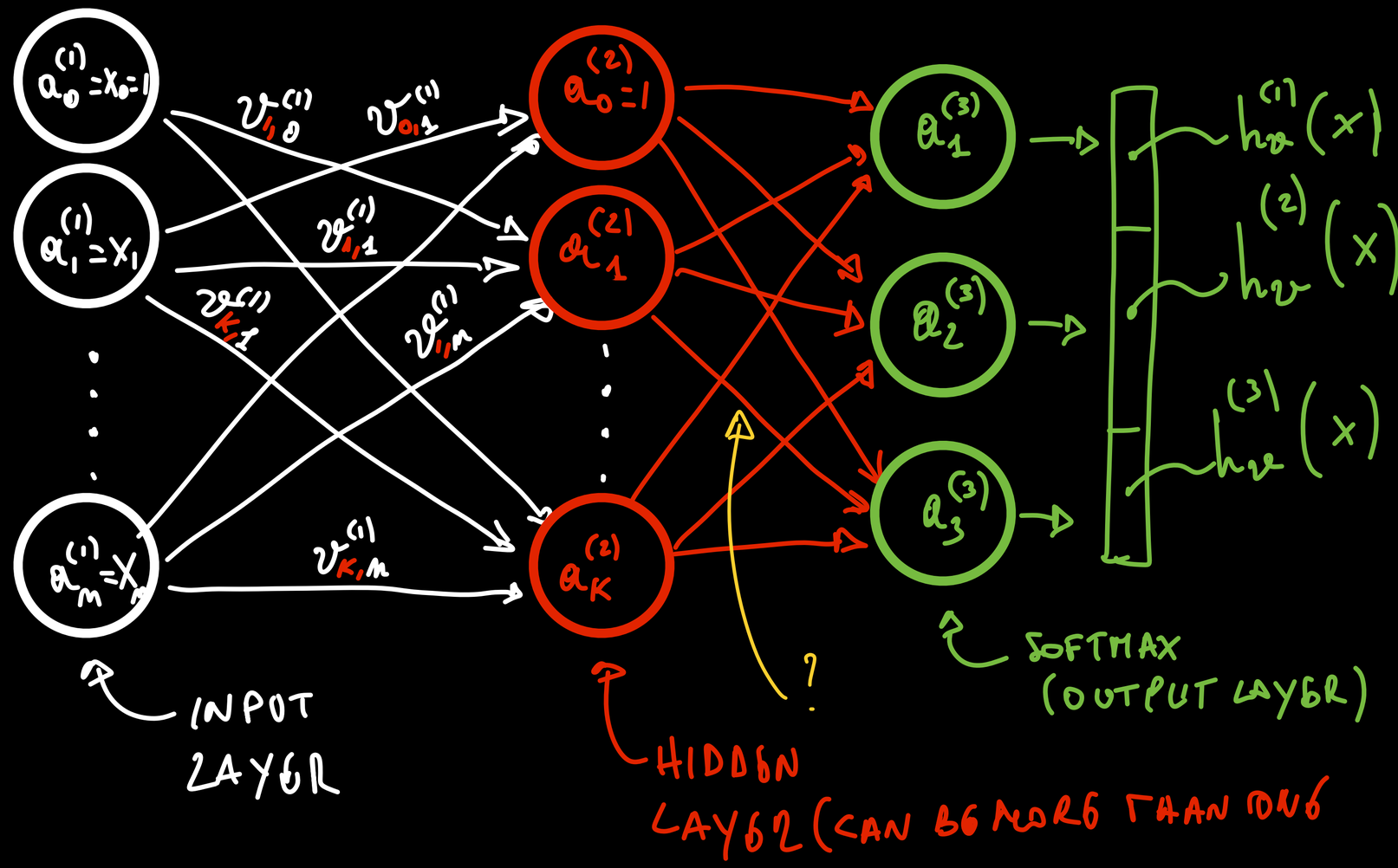
Backpropagation

(via Computational Graphs)

SEE ALSO CHAPTER 7 ("THE BACKPROPAGATION ALGORITHM")
OF THE BOOK "NEURAL NETWORKS - A SYSTEMATIC INTRODUCTION"



AL CRESCERE DELLA DIMENSIONE DELL'INPUT CRESCE LA PROBABILITÀ CHE LO SPAZIO SIA NON LINEARE (ADD MORE LAYER OR/AND MORE NEURONS PER LAYER)

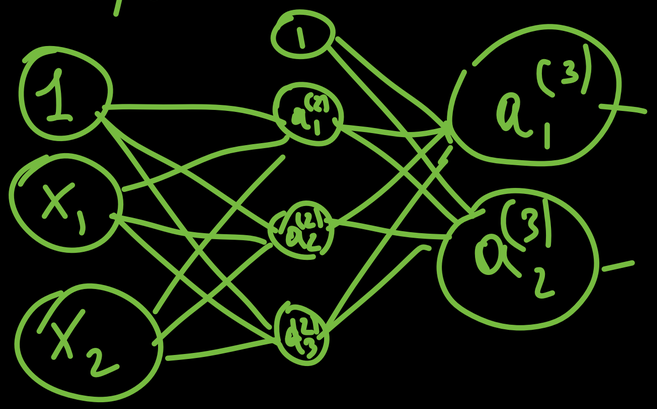
$a_0^{(1)} = x_0 = 1 \qquad a_1^{(1)} = x_1 \qquad a_2^{(1)} = x_2$ INPUT

$\Theta^{(1)} = \begin{bmatrix} \Theta_{1,0}^{(1)} & \Theta_{1,1}^{(1)} & \Theta_{1,2}^{(1)} \\ \Theta_{2,0}^{(1)} & \Theta_{2,1}^{(1)} & \Theta_{2,2}^{(1)} \\ \Theta_{3,0}^{(1)} & \Theta_{3,1}^{(1)} & \Theta_{3,2}^{(1)} \end{bmatrix}$ 3×3

$\Theta^{(2)} = \begin{bmatrix} \Theta_{1,0}^{(2)} & \Theta_{1,1}^{(2)} & \Theta_{1,2}^{(2)} & \Theta_{1,3}^{(2)} \\ \Theta_{2,0}^{(2)} & \Theta_{2,1}^{(2)} & \Theta_{2,2}^{(2)} & \Theta_{2,3}^{(2)} \end{bmatrix}$ 2×4

WITH Θ WE INDICATE ALL SET OF PARAMETERS

CONSIDERING THE MODEL ABOVE, THE NETWORK IS COMPOSED BY 2 INPUT FEATURES EXTENDED WITH BIAS, JUST ONE HIDDEN LAYER WITH 3 NEURONS + THE BIAS NEURON OF THE HIDDEN LAYER AND AN OUTPUT LAYER WITH 2 NEURONS



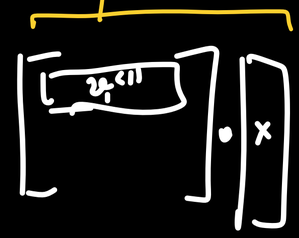
$a_0^{(2)} = 1$

x_i

QUANTI PARAMETRI HA LA NETWORK NELLA SLIDE PRECEDENTE?

$$a_1^{(2)} = g\left(\sum_{i=0}^{2} \Theta_{1,i}^{(1)} a_i^{(1)}\right) = g\left(\Theta_{1,0}^{(1)} x_0 + \Theta_{1,1}^{(1)} x_1 + \Theta_{1,2}^{(1)} x_2^{(1)}\right)$$

↑ ACTIVATION FUNCTION (e.g., SIGMOID, RELU, ...)



$$a_2^{(2)} = g\left(\sum_{i=0}^{2} \Theta_{2,i}^{(1)} a_i^{(1)}\right) \qquad a_3^{(2)} = g\left(\sum_{i=0}^{2} \Theta_{3,i}^{(1)} a_i^{(1)}\right)$$

$$a_1^{(3)} = g\left(\sum_{i=0}^{3} \Theta_{1,i}^{(2)} a_i^{(2)}\right) \qquad a_2^{(3)} = g\left(\sum_{i=0}^{3} \Theta_{2,i}^{(2)} a_i^{(2)}\right)$$

$$h_\Theta = \left[a_1^{(3)}, a_2^{(3)}\right]^T$$

SE LA NETWORK HA S_ℓ UNITÀ NEL LAYER ℓ E $S_{\ell+1}$ UNITÀ NEL LAYER $\ell+1$, ALLORA $\Theta^{(\ell)}$ HA UNA DIMENSIONE DI $S_{\ell+1} \times S_\ell$

(NOTA CHE ① È CONTEGGIATO COME NEURONE)

UNA VOLTA FISSATI I PARAMETRI ϑ, LE FORMULE NELLA SLIDE PRECEDENTE POSSONO ESSERE USATE PER OTTENERE LA DISTRIBUZIONE DI PROBABILITÀ SULLE VARIE CLASSI DI UN NUOVO DATO DI INPUT x.

$$h_\vartheta(x) = \begin{bmatrix} a_1^{(L)} \\ a_2^{(L)} \\ \vdots \\ a_K^{(L)} \end{bmatrix} = \begin{bmatrix} P(y=1|x) \\ P(y=2|x) \\ \vdots \\ P(y=3|x) \end{bmatrix}$$

$L \equiv$ # LAYER DELLA RETE $\equiv$ ULTIMO LAYER

$$\sum_{i=1}^{K} a_i^{(L)} = 1$$

QUESTA OPERAZIONE PRENDE IL NOME DI FEEDFORWARD PROPAGATION (I DATI SI PROPAGANO NEL GRAFO CHE RAPPRESENTA LA RETE DA SINISTRA VERSO DESTRA).

PER DETERMINARE I PARAMETRI BISOGNA USARE GRADIENT DESCENT PER OTTENERE IL MINIMO DELLA FUNZIONE COSTO

$$J(\vartheta) = \frac{1}{2m} \sum_{i=1}^{m} L\left(\vartheta, x^{(i)}, y^{(i)}\right) + \frac{\lambda}{2} \sum_{l=1}^{L-1} \sum_{i=1}^{S_l} \sum_{j=1}^{S_{l+1}} \left(\vartheta_{j,i}^{(l)}\right)^2$$

L = LOSS FUNCTION (CLASSIFICATION/REGRESSION) CROSS-ENTROPY MSE

GRADIENT DESCENT NEEDS THE COMPUTATION OF THE DERIVATIVE OF $J(\theta)$

$$\theta_{ji}^{(\ell)} \leftarrow \theta_{ji}^{(\ell)} - \alpha \frac{\partial J(\theta)}{\partial \theta_{ji}^{(\ell)}}$$

COMPUTE THE DERIVATIVE FOR EVERY PATH OF THE NETWORK IS INEFFICIENT !!!

MANUAL CODING IS BORING!

WE NEED A SMART ALGORITHM → BACKPROPAGATION

FEEDFORWARD NETWORK

is a

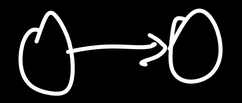
COMPUTATIONAL GRAPH

NEURAL NETWORK REPRESENTS
A CHAIN OF FUNCTION COMPOSITIONS
WHICH TRANSFORM AN INPUT
TO AN OUTPUT

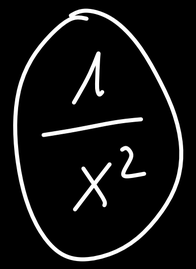
ACTIVATION FUNCTIONS
ARE ASSUMED SAME FOR EACH NODE

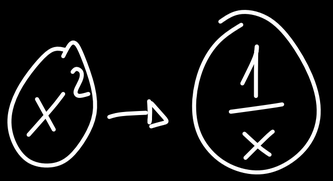
NODES ARE COMPUTING UNITS

EDGES TRANSMIT (WEIGHT)
NUMERICAL INFORMATION
FROM NODE TO NODE
(FORWARD)



EACH COMPUTING UNIT
IS ABLE TO COMPUTE
A SINGLE
PRIMITIVE
FUNCTION OF
ITS INPUT



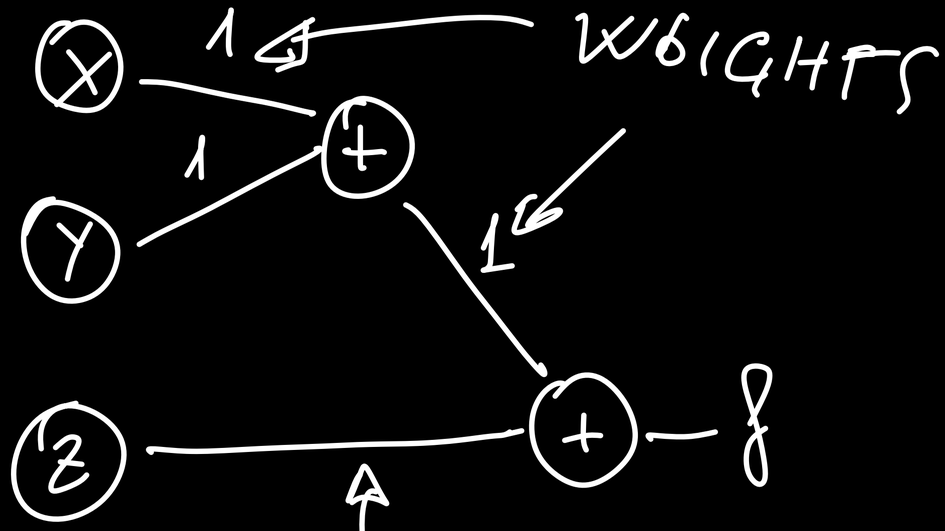


PRIMITIVE FUNCTIONS
HAVE TO BE CONTINUOUS
AND DIFFERENTIABLE

EXAMPLE OF COMPUTATIONAL GRAPH TO WICH WE KNOW ANALITICAL FORM IN ADVANCE:

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

PRIMITIVE FUNCTION



WHEN WEIGHT IS NOT PRESENT VALUE IS 1

LEARNING PROBLEM

- FIND THE OPTIMAL SET OF WEIGHTS SO THAT A NEURAL NETWORK h_w APPROXIMATE A GIVEN FUNCTION F AS CLOSELY AS POSSIBLE WITH RESPECT TO A LOSS FUNCTION

- F IS NOT GIVEN EXPLICITLY, BUT ONLY IMPLICITLY THROUGH EXAMPLES (TRAINING SET)

- WEIGHTS ARE SMALL REAL NUMBERS INITIALIZED RANDOM AND CONSIDERING SIMMETRY BREAKING

BACKPROPAGATION

IT IS USED TO FIND THE LOCAL MINIMA OF THE LOSS FUNCTION (e.g., THE WEIGHTS RELATED TO MSE LOCAL MINIMA) CONSIDERING THE TRAINING SET:

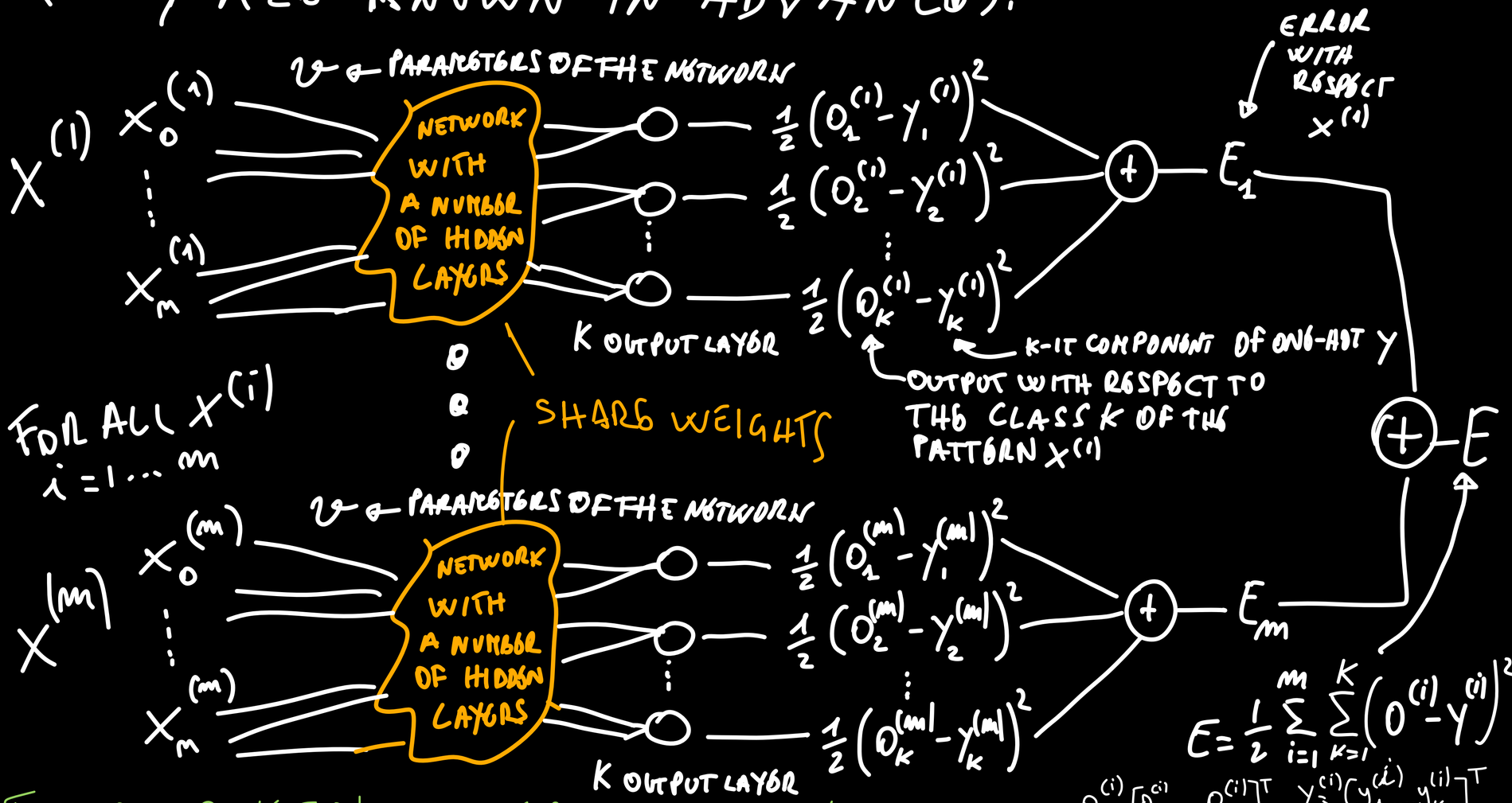
1) THE NETWORK IS INITIALIZED WITH RANDOM WEIGHTS ← REMIND SYMMETRY BREAKING
2) EVERY TRAINING INPUT IS FEED TO THE NETWORK
3) FOR EVERY TRAINING INPUT THE GRADIENT OF THE LOSS FUNCTION IS COMPUTED WITH RESPECT TO THE PARAMETERS OF THE NETWORK
4) THE COMPUTED GRADIENTS ARE CUMULATED AND THE AVERAGE IS COMPUTED
5) THE GRADIENT OF THE LOSS COMPUTED AT 4) IS USED TO UPGRADE WEIGHTS WITH GRADIENT DESCENT
6) RESTART FROM 2) FOR AGIVEN NUMBER OF EPOCHS (ITERATIONS)

CONVERGENCE IS MONITORED LOOKING AT THE PLOT OF THE LOSS FUNCTION

(SAVE WEIGHTS EVERY TIME YOU FIND A NEW MINIMUM. YOU CAN RESTART FROM THERE)

EXTENDED NETWORK (MULTIPLE REGRESSION EXAMPLES) ≡ MSE LOSS

IT IS THE NETWORK WHICH INCLUDES THE LOSS FUNCTION AND IT IS EXTENDED FOR ALL THE TRAINING SAMPLES (THEY ARE KNOWN IN ADVANCE).



DESIDERATA: SUPPOSE WE HAVE P PARAMETERS IN THE NET $\vartheta_1 \dots \vartheta_P$ (CONSIDERING ALL LAYERS). WE WISH TO COMPUTE:

$$\nabla E = \left(\frac{\partial E}{\partial \vartheta_1}, \frac{\partial E}{\partial \vartheta_2}, \dots, \frac{\partial E}{\partial \vartheta_P} \right)$$

SPECIFICALLY, WE NEED THE FOLLOWING VALUE TO UPGRADE THE PARAMETERS WITH GRADIENT DESCENT:

$$\Delta \vartheta_p = -\alpha \frac{\partial E}{\partial \vartheta_p} \quad \forall p = 1 \dots P$$

α: LEARNING RATE

CONSIDERING THE EXTENDED NETWORK WE CAN ONLY MODIFY PARAMETERS TO REDUCE ERRORS (i.e., ALL THE OTHER VALUES ARE FIXED)

BACKPROPAGATION EXPLOITS CHAIN RULE:

IF $F(x) = f(g(x))$ THEN $F'(x) = g'(x) f'(g(x))$

DERIVATIVE OF NETWORK FUNCTION

GOAL : FIND A SMART WAY TO COMPUTE GRADIENT

OBSERVATION 1: NETWORK IS A COMPOSITION OF DERIVABLE FUNCTIONS

OBSERVATION 2 : WE KNOW THE CHAIN RULE PROPERTY

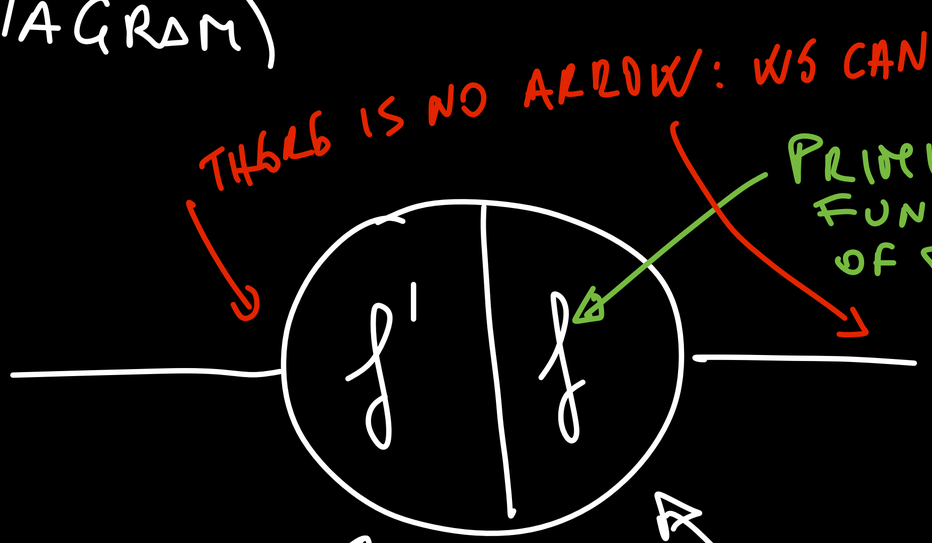
$$f(g(x)) \xrightarrow{\text{DERIVATE}} g'(x) f'(g(x))$$

TO FIND A SMART WAY TO COMPUTE GRADIENT ON A NETWORK WE WILL CONSIDER A VERY OLD CONCEPT...

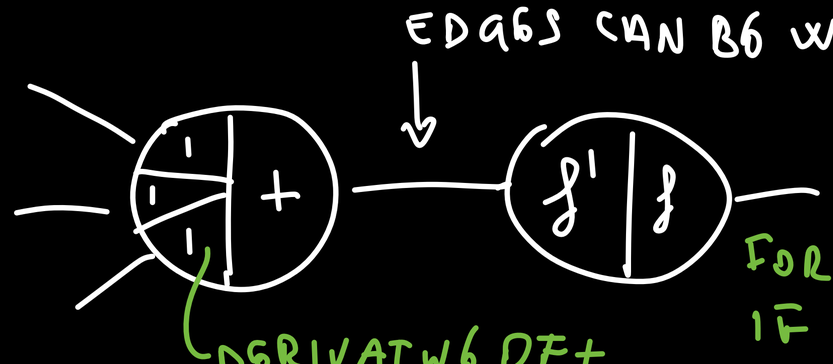
COMPUTATIONAL GRAPHS!!

COMPOSITE STRUCTURE OF THE NODES

- WE GIVE A COMPOSITE STRUCTURE TO EACH NODE OF THE NETWORK $\Rightarrow$ BACKPROPAGATION DIAGRAM (B-DIAGRAM)



CONSIDERING PRIMITIVE FUNCTION, INTEGRATION FUNCTION (SUM) ARE SEPARATED FROM THE ACTIVATION FUNCTIONS



DERIVATIVE OF + IS 1 FOR EACH OF THE LOCAL GRADIENT

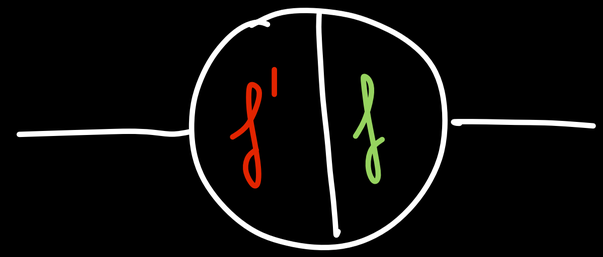
FOR EXAMPLE IF f IS THE SIGMOID FUNCTION

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

THE DERIVATIVE f' IS $\sigma(x)(1-\sigma(x))$

THE NETWORK IS EVALUATED IN TWO STEPS FOR TRAINING:

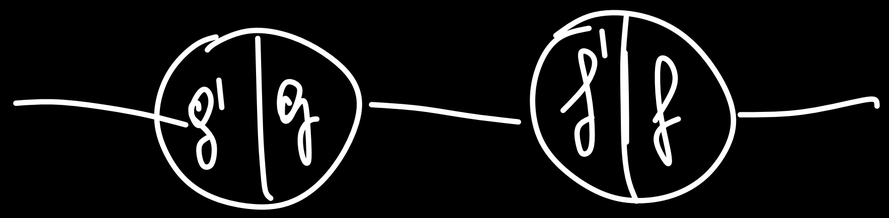
1) FEEDFORWARD STEP: INFORMATION GOES FROM LEFT TO RIGHT
TO EACH NODE THE PRIMITIVE FUNCTIONS ARE EVALUATED
AS WELL AS THE DERIVATIVES (OF THE LEFT SIDE).
RESULTS ARE STORED. THE RESULTS OF THE RIGHT SIDE
ARE FORWARDED TO THE CONNECTED UNITS ON THE RIGHT



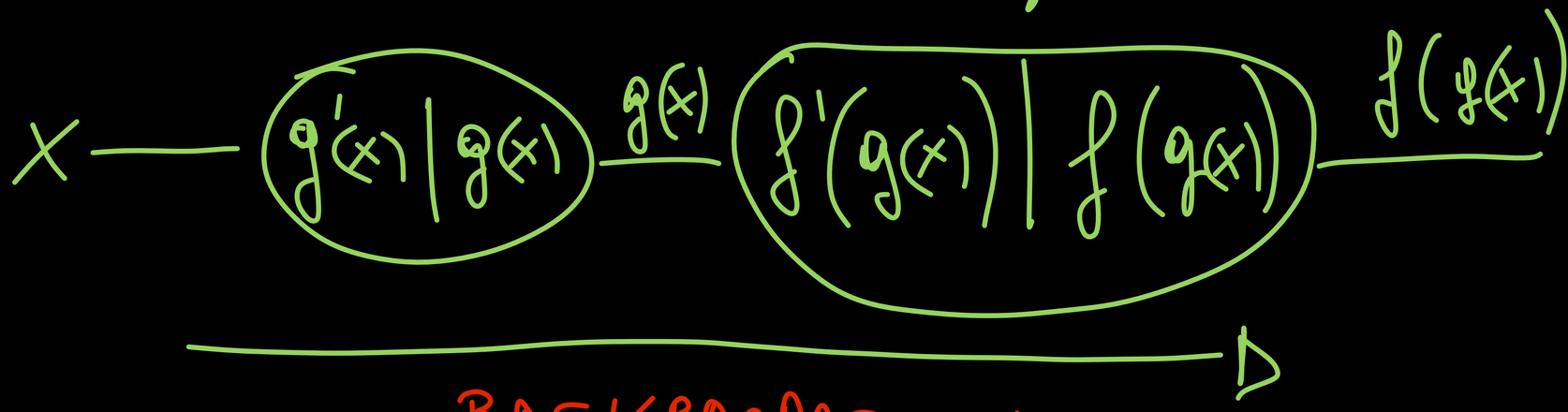
2) BACKWARD STEP: IT IS USED TO EVALUATE ERRORS
AND TO PROPAGATE THEM BACKWARD TO UPGRADE THE
PARAMETERS. IN THIS CASE THE RESULTS OF THE LEFT
SIDE (DERIVATIVES) ARE USED. THERE ARE THREE CASES
TO BE CONSIDERED (WE WILL SEE THEM IN THE FOLLOWING SLIDES).

ANY COMPUTATIONAL GRAPH (FUNCTION COMPOSITION)
CAN BE EVALUATED WITH BACKPROPAGATION TO
COMPUTE DERIVATIVES

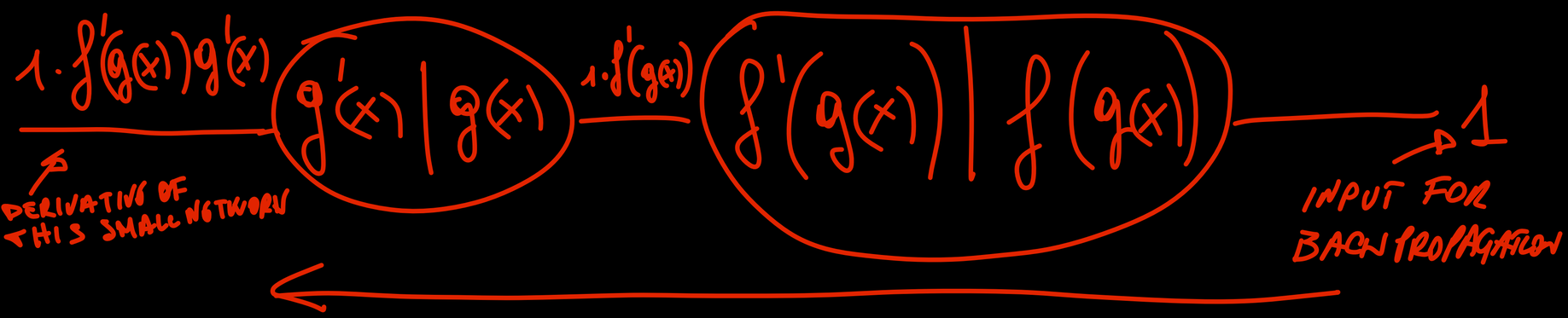
FUNCTION COMPOSITION (CASE 1)



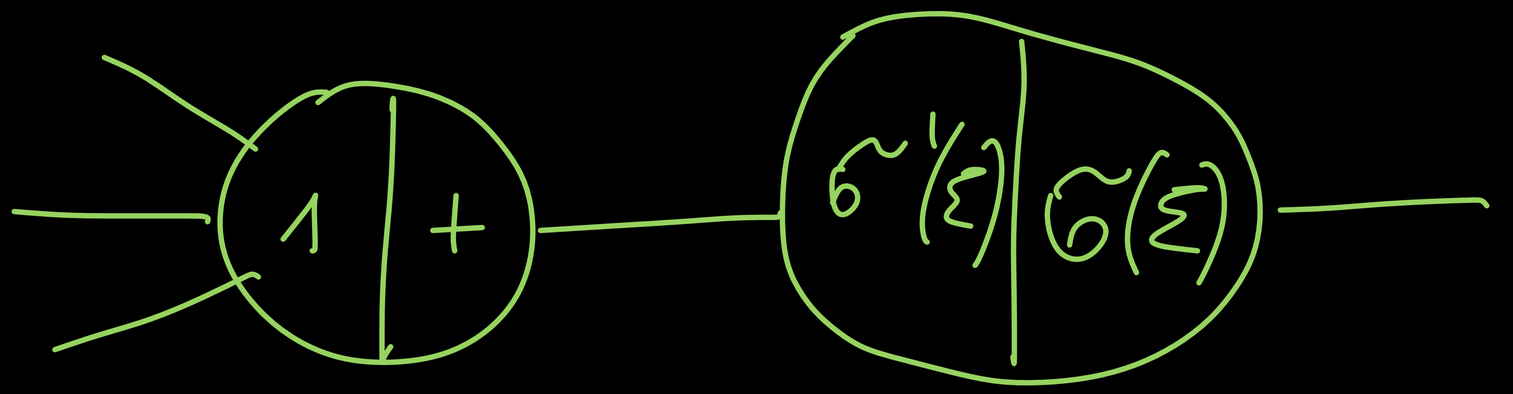
FEEDFORWARD PASS/STEP



BACKPROPAGATION



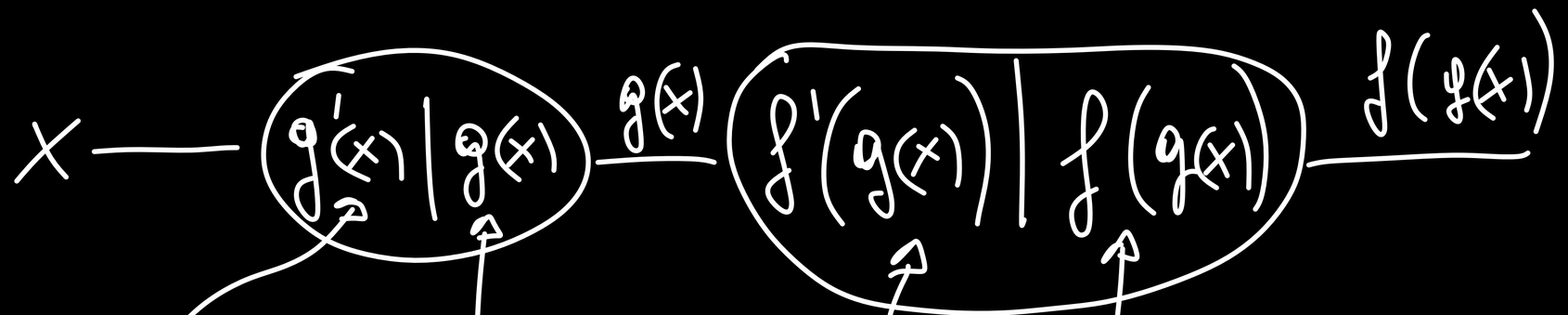
CONSIDERING SIGMOID WHERE INPUT OF IT IS A LINEAR COMBINATION



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

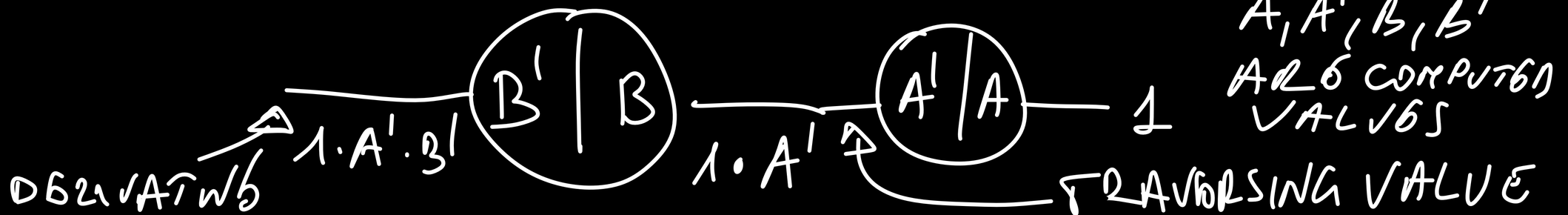
IN THIS CASE z IS A LINEAR COMBINATION Σ

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

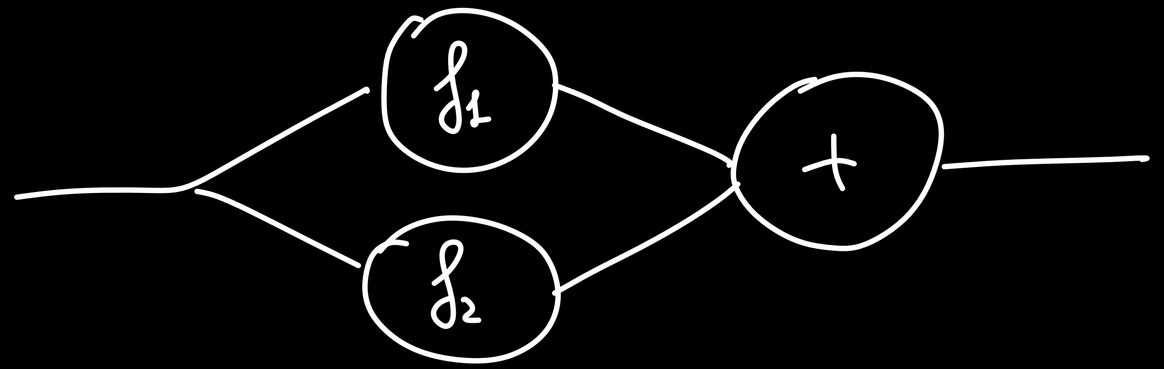


- BACKPROPAGATION PROVIDE A WAY TO IMPLEMENT THE CHAIN RULE PRINCIPLE
  - INPUT FROM RIGHT SIDE IS ALWAYS 1

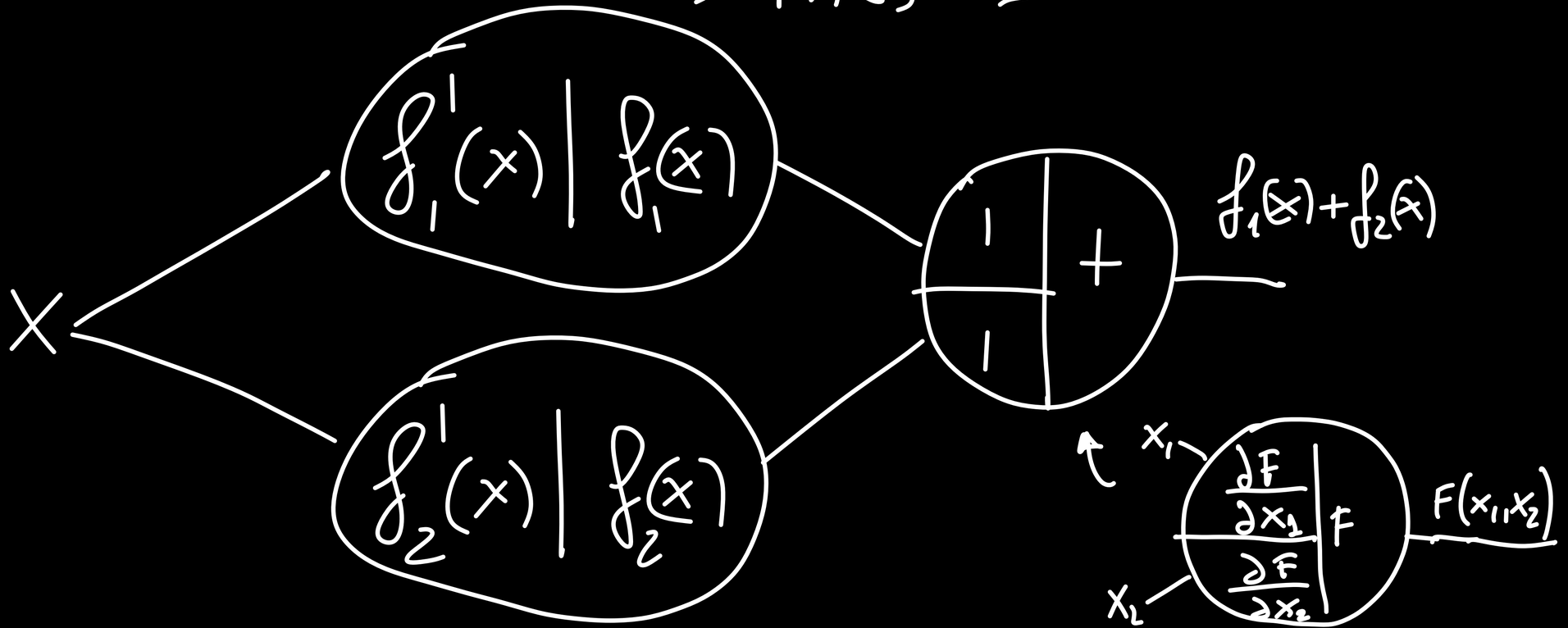
IN GENERAL:



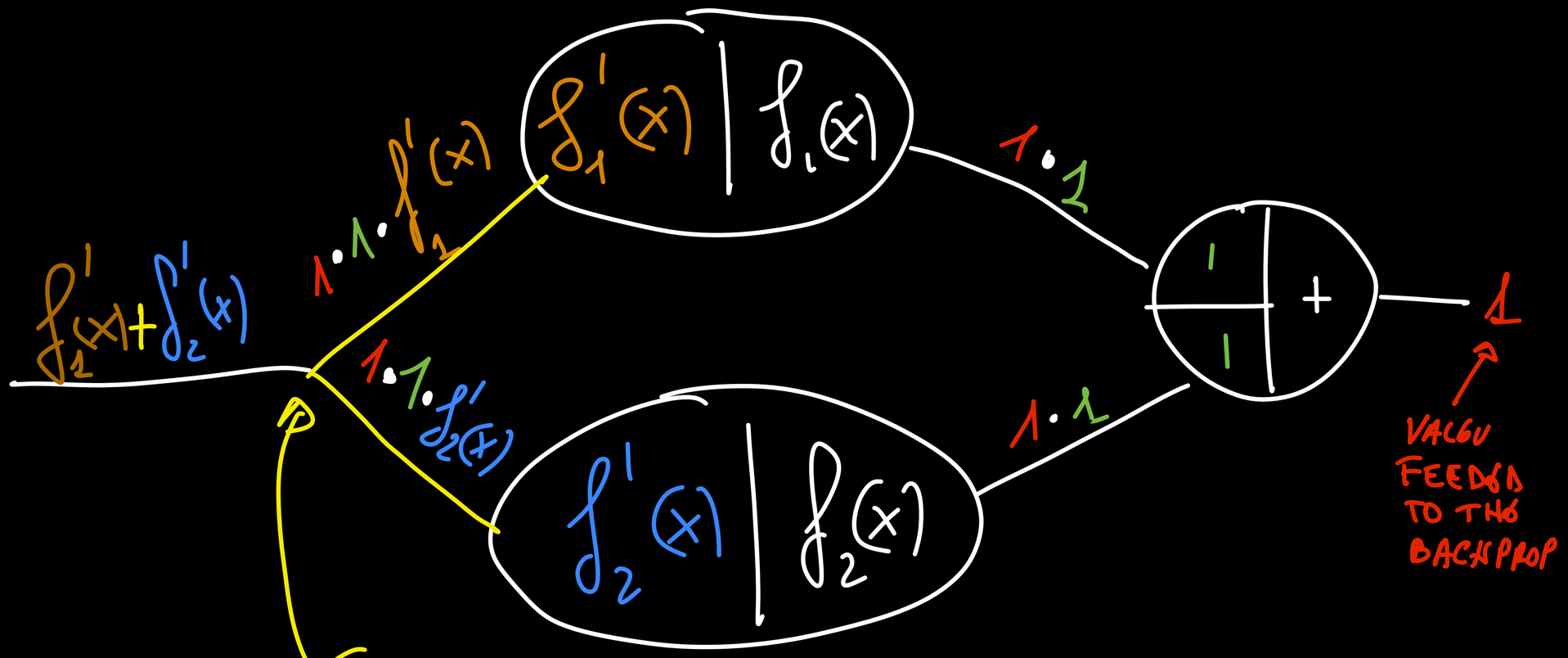
FUNCTIONS ADDITION (CASE 2)



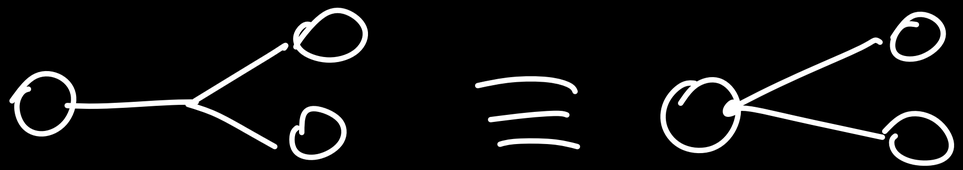
FORWARD PASS →



BACKWARD PASS ←



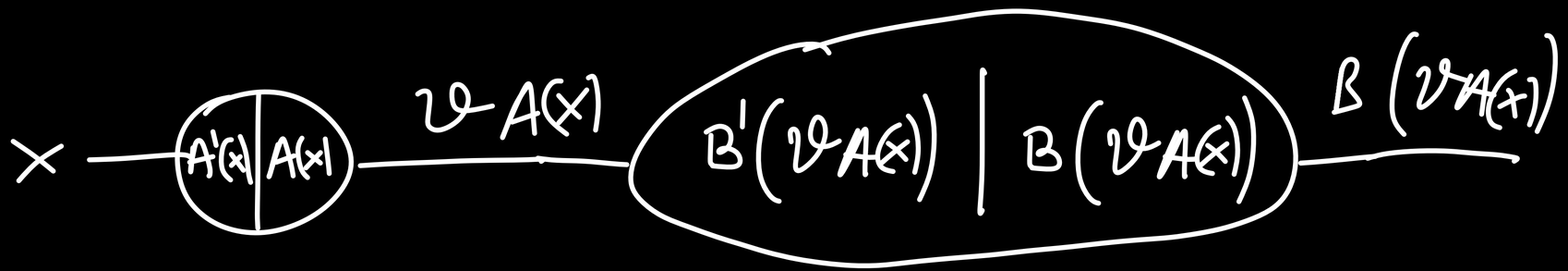
GOING BACKWARD THE VALEUS ARE SUMMED WHEN COMING FROM THE SAME BRANCH (NODE)



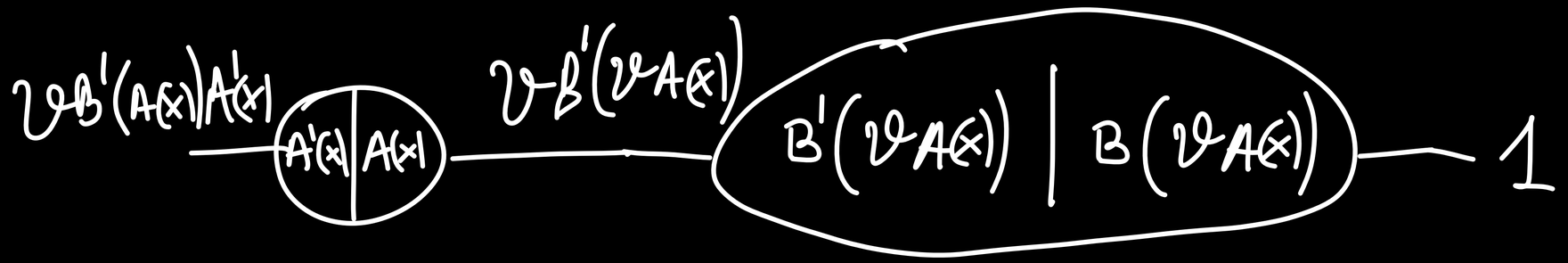
WEIGHTED EDGES (CASE 3)

WEIGHTS ARE MULTIPLIED FORWARD AND BACKWARD

FORWARD →



BACKWARD ←



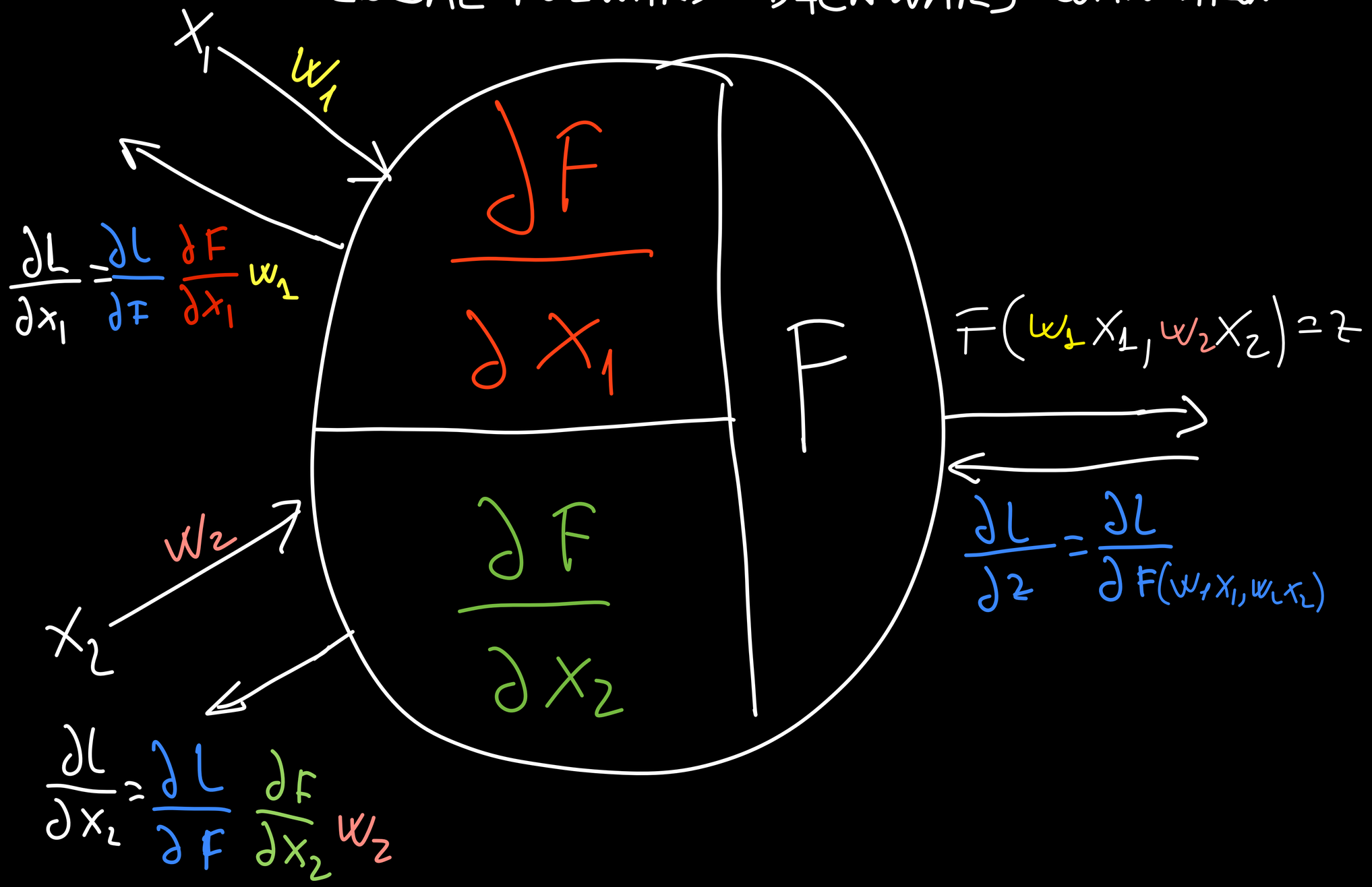
WEIGHTS MODULATE TRANSMITTED VALUES IN BOTH DIRECTIONS

CONSIDER A NETWORK F WITH AN INPUT x,
THE DERIVATIVE IS COMPUTED AS FOLLOWING:

1) FORWARD PASS: THE INPUT IS EVALUATED FROM LEFT TO RIGHT CONSIDERING THE NETWORK. VALUES RELATED PRIMITIVE FUNCTIONS AND THEIR DERIVATIVES ARE COMPUTED AND STORED

2) BACKWARD PASS: THE CONSTANT 1 IS IMPOSED AS RIGHT INPUT OF THE NETWORK. THE NETWORK IS COMPUTED FROM RIGHT TO LEFT. INFORMATION WHICH ARRIVE TO A GIVEN NODE ARE SUMMED AND THE RESULT MULTIPLIED WITH THE VALUE SAVED ON THE LEFT PART OF THE NET AND WITH THE PARAMETER OF THE EDGE. THE RESULT IS THE DERIVATIVE OF THE FUNCTION F WITH RESPECT TO x

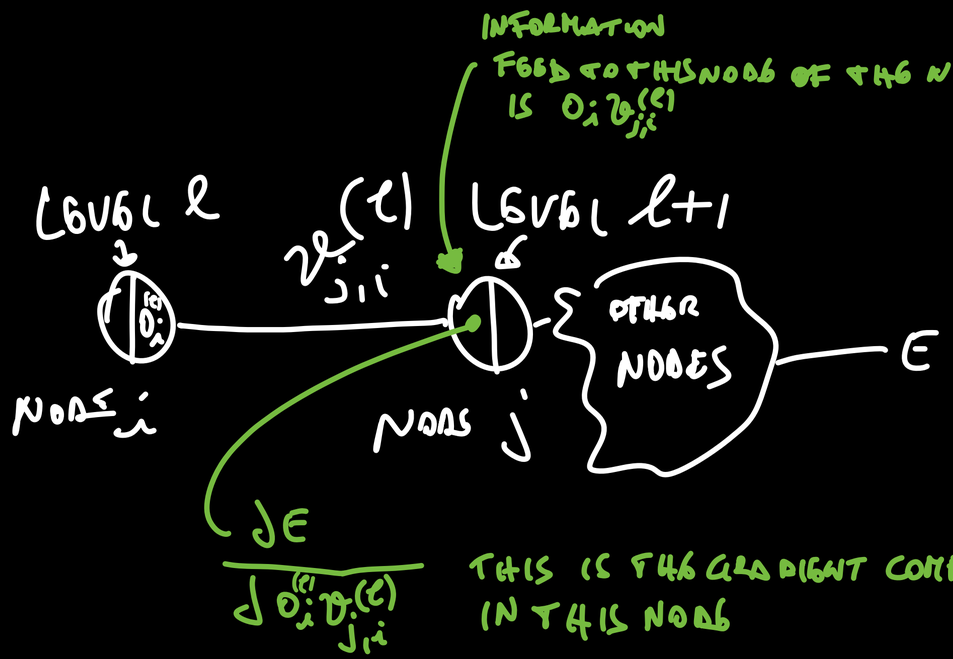
LOCAL FORWARD - BACKWARD COMPUTATION

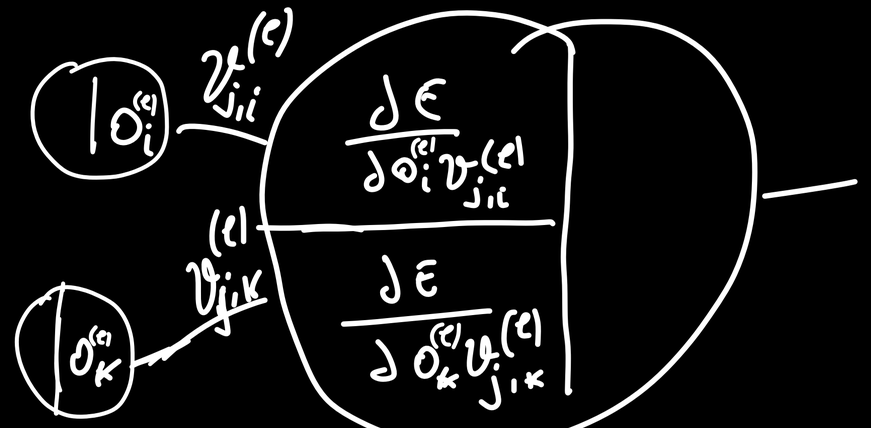


WITH REGARDS TO THE EXTENDED NETWORK WE WANT TO COMPUTE THE FOLLOWING FOR EACH PARAMETER $v_{ji}^{(l)}$

$$\frac{\partial E}{\partial v_{ji}^{(l)}}$$

$v_{ji}^{(l)}$: LEVEL l, FROM NODE i TO NODE j





NOTE THAT $o_i^{(\ell)}$ IS ALREADY COMPUTED DURING BACKPROPAGATION.

HENCE IT IS A CONSTANT DURING THE BACKWARD PASS AND ALLOWS TO WRITE THE FOLLOWING:

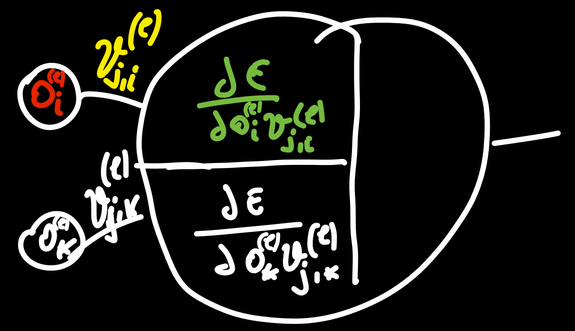
$$\frac{\partial E}{\partial w_{ji}^{(\ell)}} = \frac{\partial o_i^{(\ell)} w_{ji}^{(\ell)}}{\partial w_{ji}^{(\ell)}} \frac{\partial E}{\partial o_i^{(\ell)} w_{ji}^{(\ell)}} = o_i^{(\ell)} \frac{\partial E}{\partial o_i^{(\ell)} w_{ji}^{(\ell)}}$$

OUR DESIDERATA

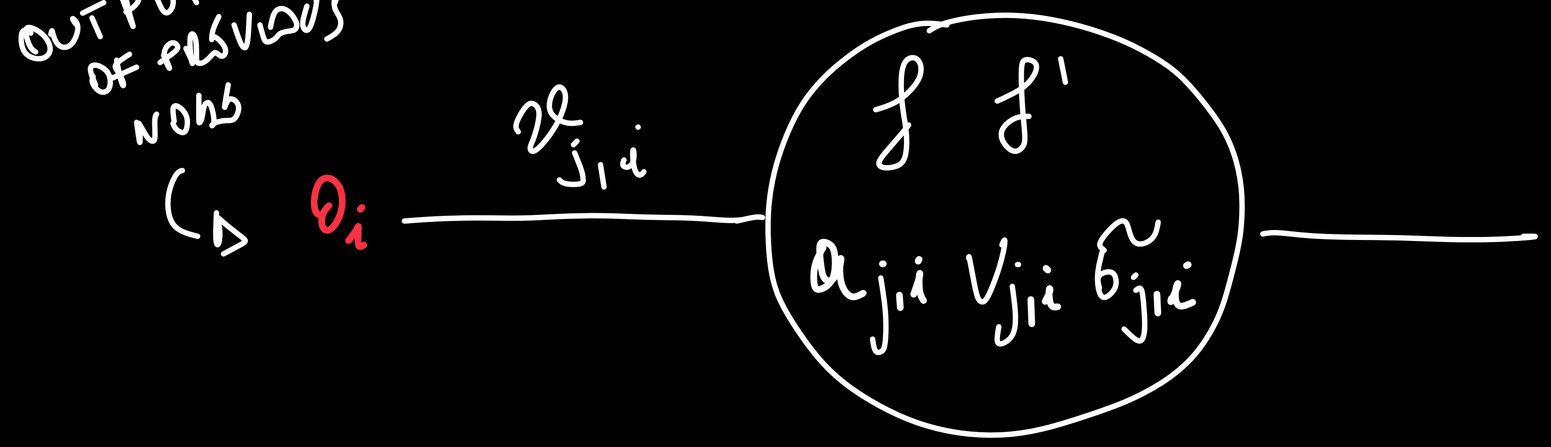
FOR THE CHAIN RULE

BACKPROPAGATION ERROR AT NODE j

(USUALLY DENOTED WITH $\tilde{\delta}_{ji}^{(\ell)}$)



FOR EACH PARAMETER WE HAVE



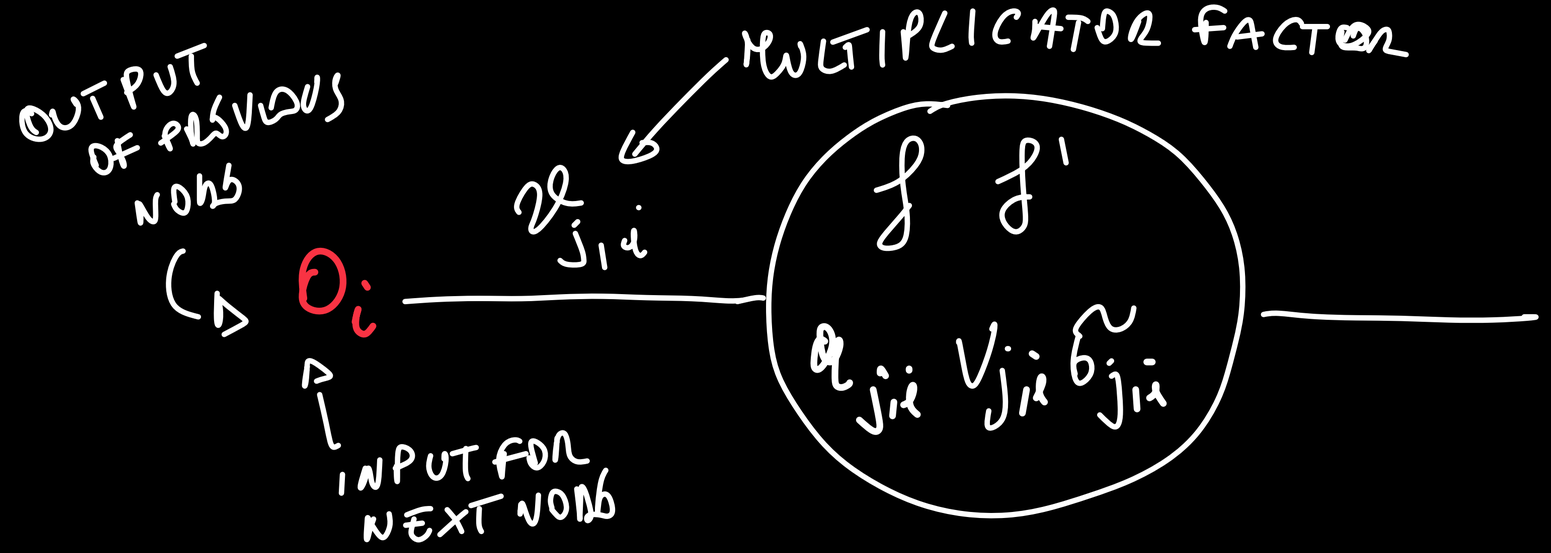
f: PRIMITIVE FUNCTION TO BE COMPUTED ON THE INPUT $O_i v_{j,i}$

f': DERIVATIVE OF THE PRIMITIVE FUNCTION TO BE COMPUTED ON THE INPUT $O_i v_{j,i}$

$a_{j,i}$: VALUE OF THE PRIMITIVE FUNCTION f COMPUTED ON $O_i v_{j,i}$ IN THE FEED FORWARD PASS

$V_{j,i}$: VALUE OF THE DERIVATIVE PRIMITIVE FUNCTION f' COMPUTED ON THE VALUE $O_i v_{j,i}$ DURING FEED FORWARD STEP

$\tilde{G}_{j,i}$: CUMULATIVE RESULT OF BACK WARD COMPUTATION WHICH IS THE DERIVATIVE OF THE ERROR RESPECT TO $O_i v_{j,i}$



THIS MEANS THAT FOR EACH NODE WE CAN EASILY COMPUTE THE GRADIENT OF THE ERROR FUNCTION E WITH RESPECT TO THE PARAMETER ASSOCIATED TO THAT NODE

$$\frac{\partial E}{\partial w_{ji}} = O_i \tilde{\sigma}_{ji} \Rightarrow$$

THIS CAN BE USED FOR GRADIENT DESCENT

$$\Delta V_{ji} = -\lambda O_i \tilde{\sigma}_{ji}$$

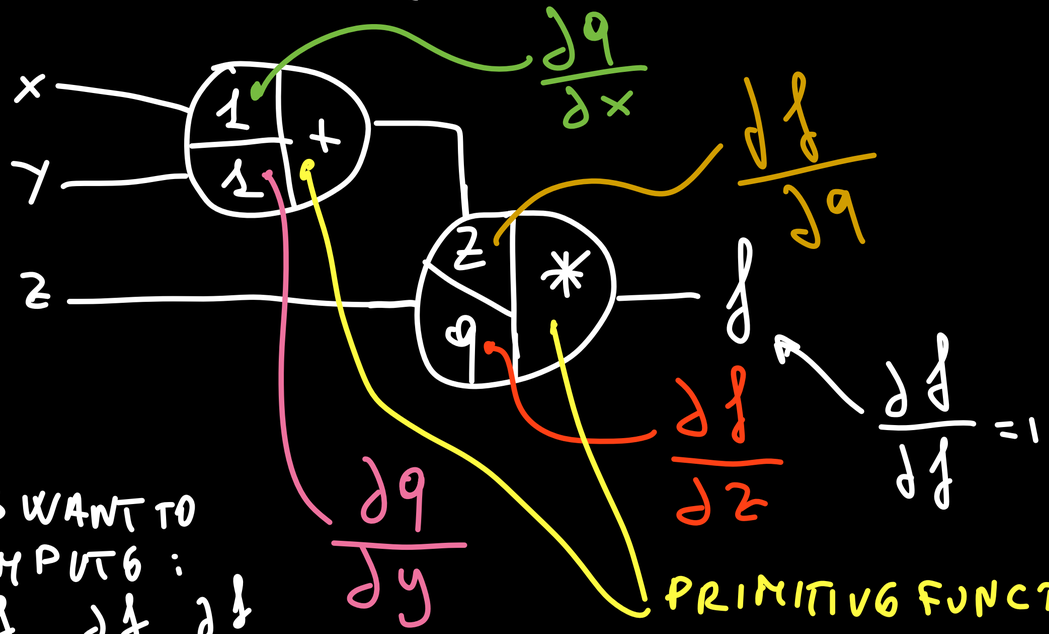
THIS CAN BE DONE FOR EACH PARAMETER OF THE NET

EXAMPLE OF FORWARD AND BACKWARD PROPAGATION

FUNCTION: $f(x, y, z) = (x+y)*z$

INPUT: $x = -2 \quad y = 5 \quad z = -4$

COMPUTATIONAL GRAPH



WE WANT TO COMPUTE: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

"SUB FUNCTIONS"

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

$f = q * z$

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

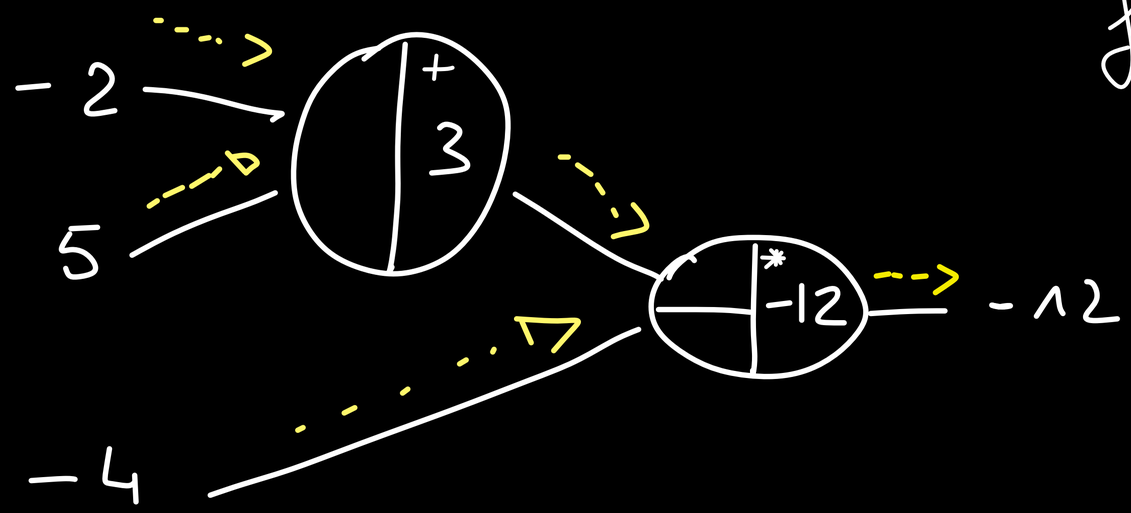
$\frac{\partial f}{\partial z} = q$

CHAIN RULE

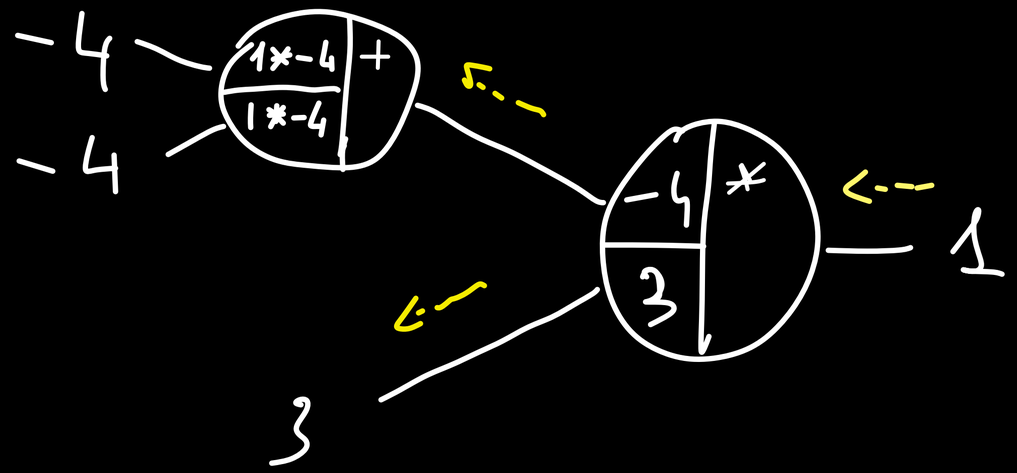
$$\frac{\partial f}{\partial x} = \frac{df}{dq}\frac{\partial q}{\partial x} = z\frac{\partial q}{\partial x} = z \cdot 1 \; ; \quad \frac{\partial f}{\partial y} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x} = z \; ; \quad \frac{\partial f}{\partial z} = q$$

FORWARD PASS ON THE VALUES $x=-2$ $y=5$ $z=-4$

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



BACKWARD PASS CONSIDERING THE ABOVE INPUT

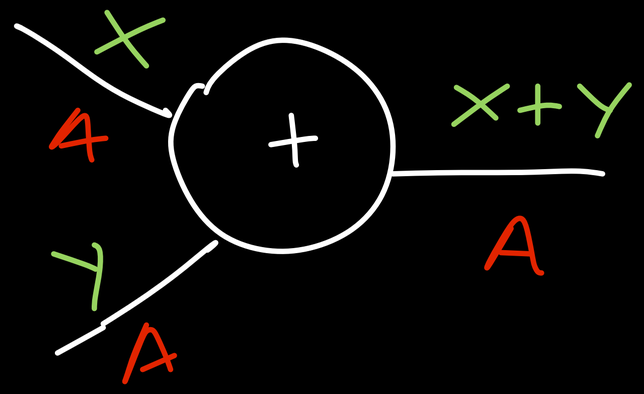


$$\frac{\partial f}{\partial x} = -4$$

$$\frac{\partial f}{\partial y} = -4$$

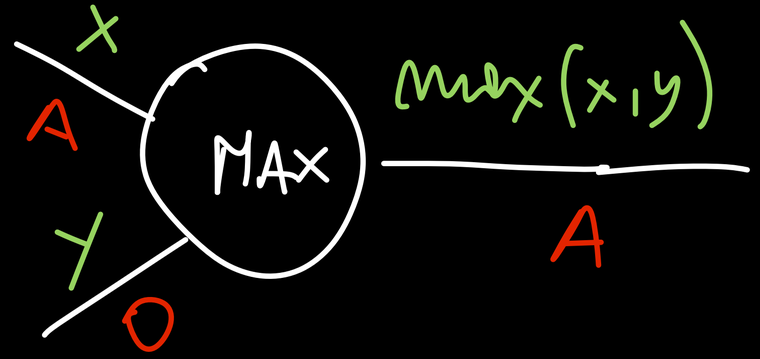
$$\frac{\partial f}{\partial z} = 3$$

ADD GATE : GRADIENT DISTRIBUTION



FORWARD
BACKWARD

MAX GATE : GRADIENT ROUTER

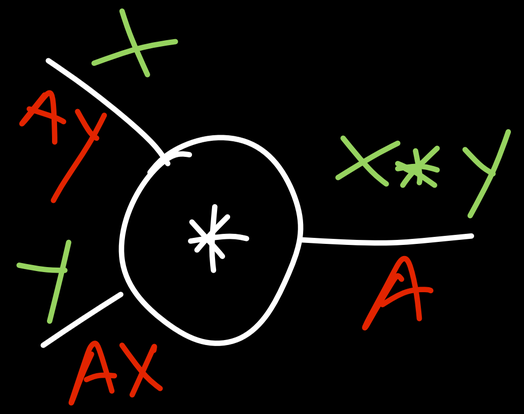


IN CASE $X > Y$

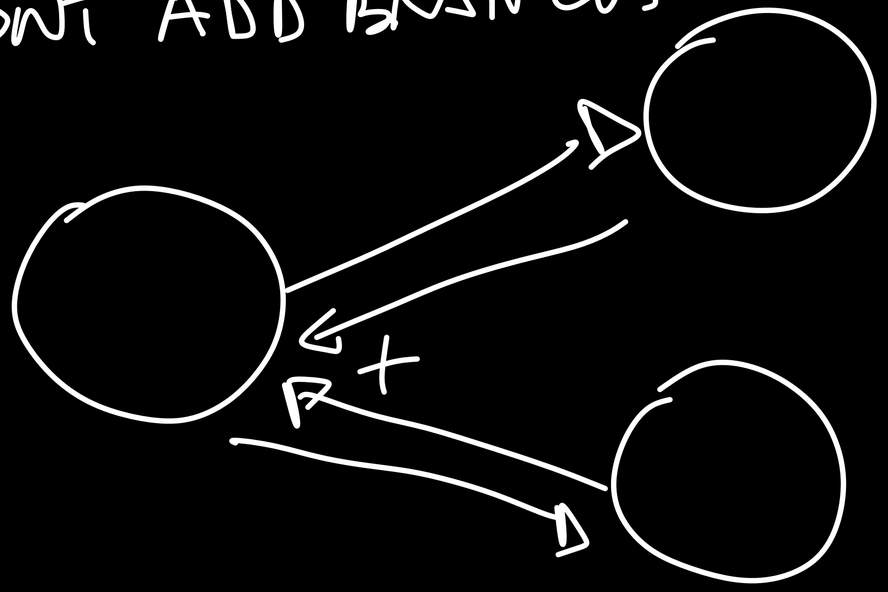
$$\max(x,y) \Rightarrow \frac{\partial \max(x,y)}{\partial x} = \mathbb{1}\{x \geq y\}$$

$$\frac{\partial \max(x,y)}{\partial y} = \mathbb{1}\{y \geq x\}$$

MUL GATE : GRADIENT SWITCH



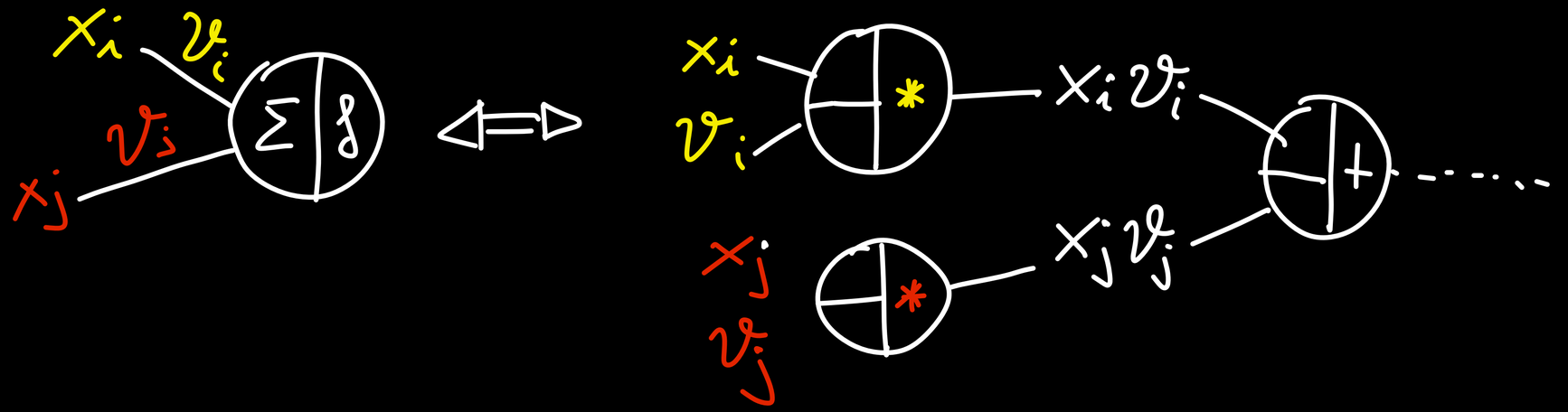
GRADIENT ADD BRANCHES



EXAMPLE WITH WEIGHTS – LOGISTIC REGRESSION

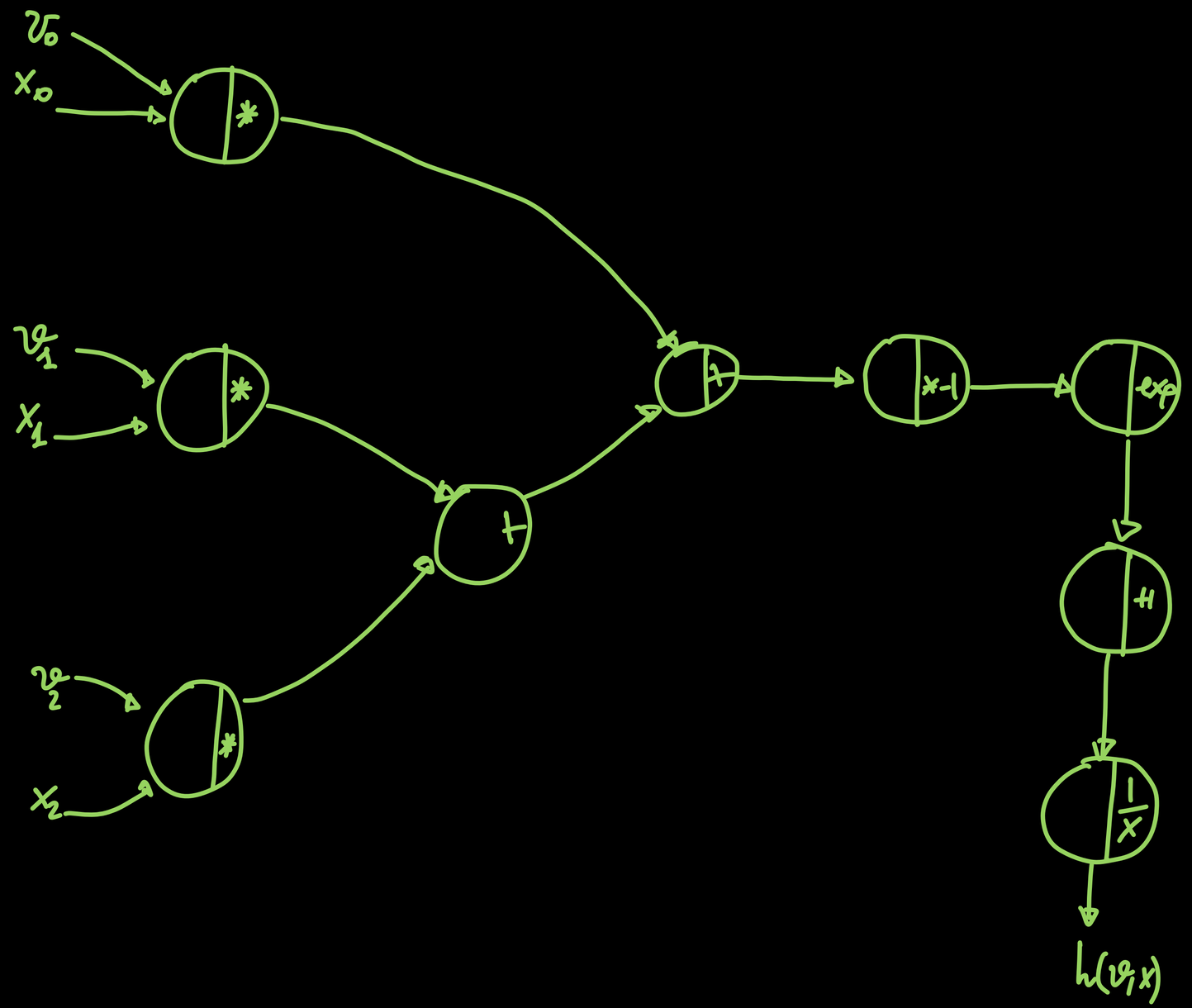
$$h(v, x) = \frac{1}{1 + e^{-(v_0 x_0 + v_1 x_1 + v_2 x_2)}}$$

$$v = \begin{bmatrix} -3 \\ -3 \\ 2 \end{bmatrix} \quad x = \begin{bmatrix} 1 \\ -2 \\ -1 \end{bmatrix}$$

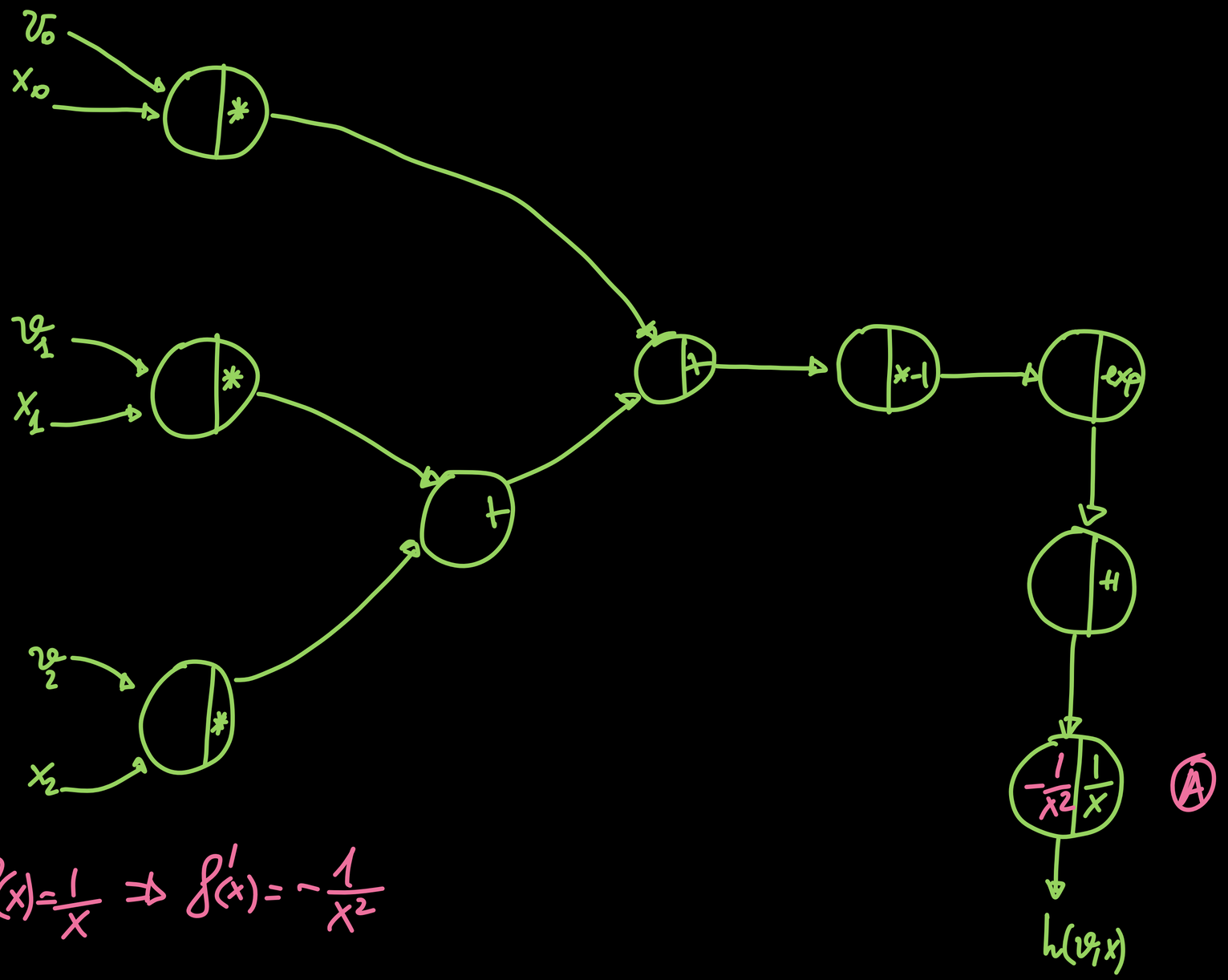


1. DRAW THE COMPUTATIONAL GRAPH
2. INSERT DERIVATIVES
3. ADD FEDFORWARD VALUES
4. ADD BACKWARD VALUES
5. COMPUTE FINAL DERIVATIVES WITH RESPECT TO v_j

① DRAW THE COMPUTATIONAL GRAPH

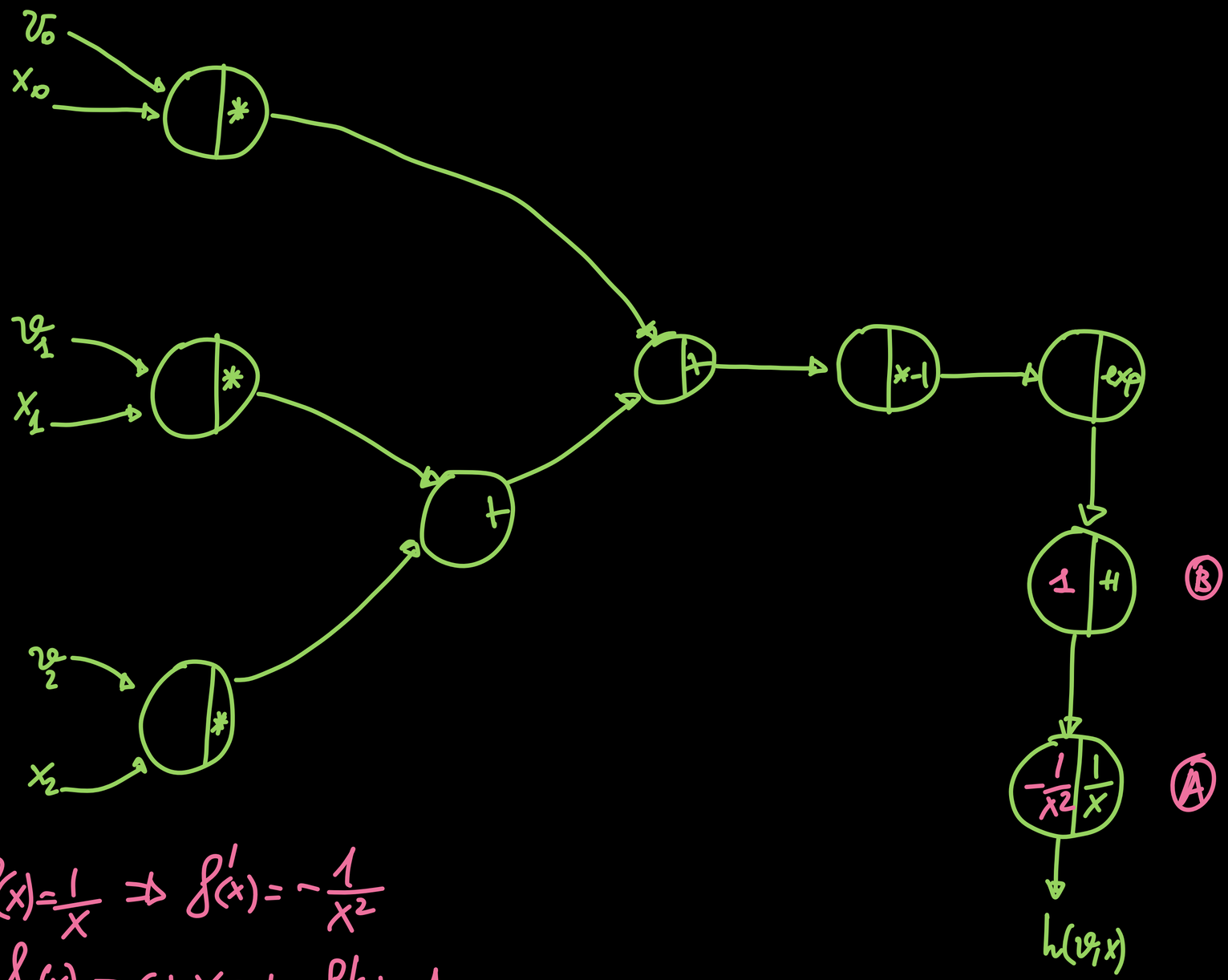


② INSERT DERIVATIVES



Ⓐ $f(x) = \frac{1}{x} \Rightarrow f'(x) = -\frac{1}{x^2}$

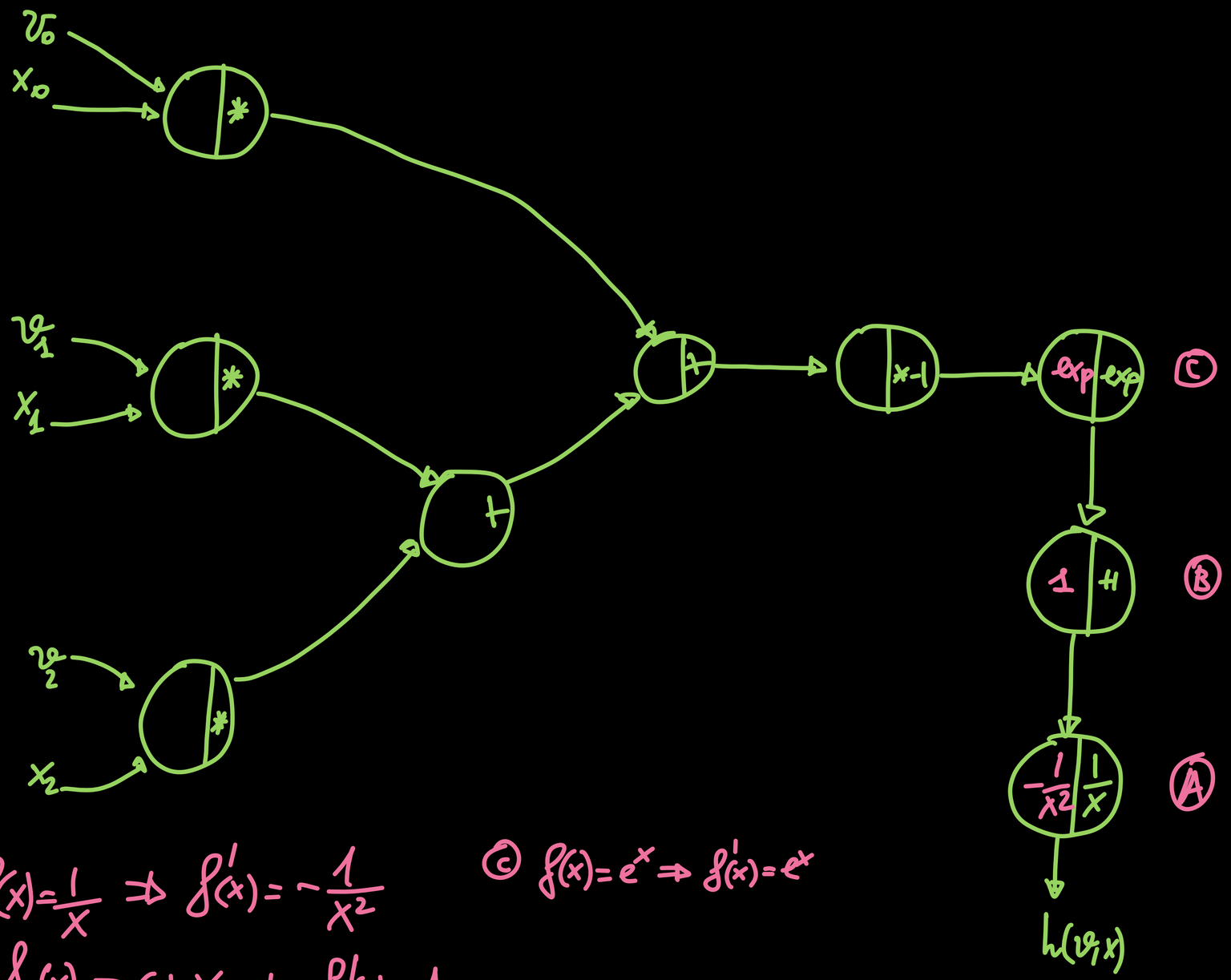
② INSERT DERIVATIVES



Ⓐ $f(x) = \frac{1}{x} \Rightarrow f'(x) = -\frac{1}{x^2}$

Ⓑ $f_c(x) = c + x \Rightarrow f'(x) = 1$

② INSERT DERIVATIVES

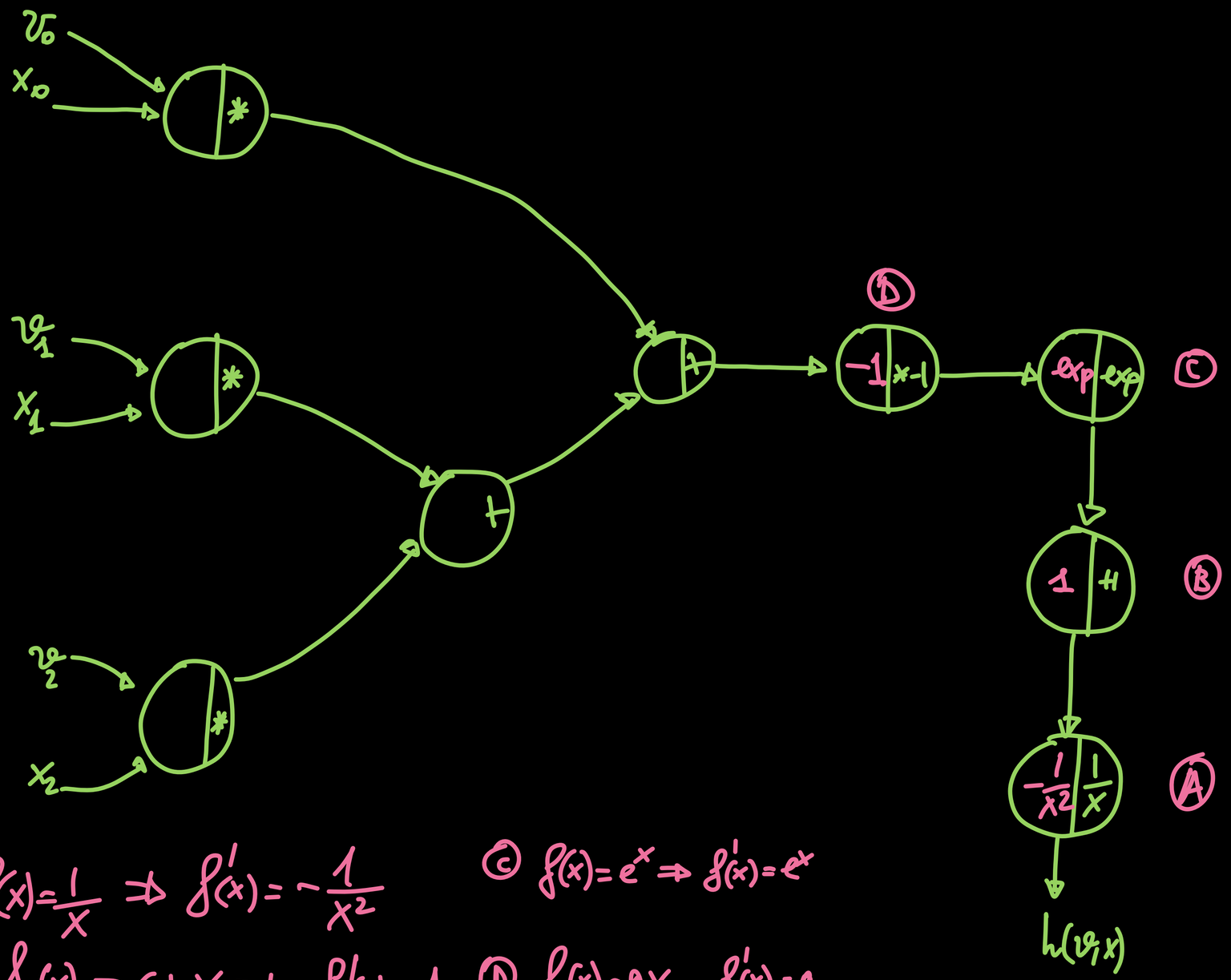


Ⓐ $f(x)=\frac{1}{x} \Rightarrow f'(x)=-\frac{1}{x^2}$

Ⓑ $f_c(x)=c+x \Rightarrow f'(x)=1$

Ⓒ $f(x)=e^x \Rightarrow f'(x)=e^x$

② INSERT DERIVATIVES



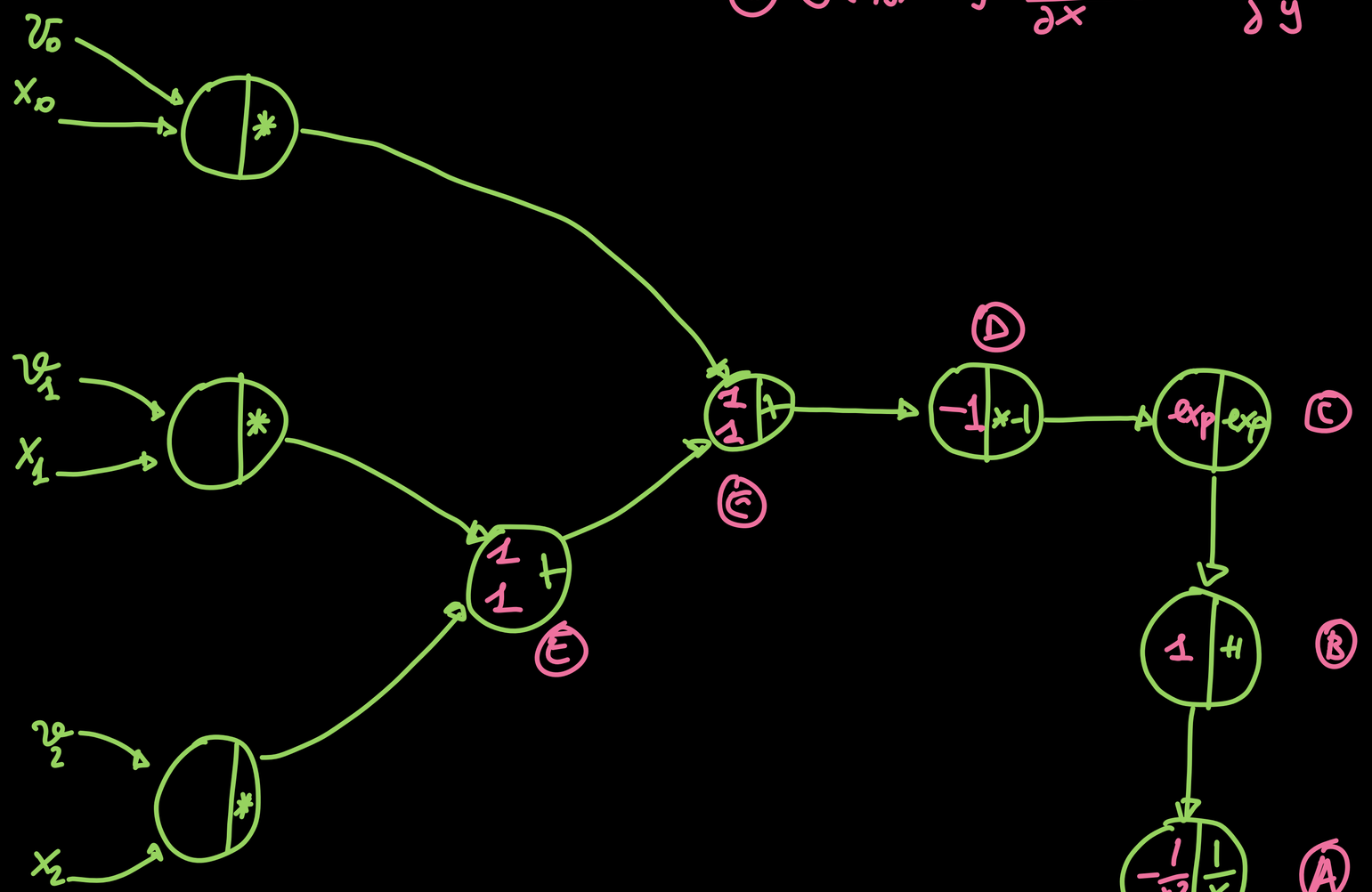
Ⓐ $f(x)=\frac{1}{x} \Rightarrow f'(x)=-\frac{1}{x^2}$

Ⓑ $f_c(x)=c+x \Rightarrow f'(x)=1$

Ⓒ $f(x)=e^x \Rightarrow f'(x)=e^x$

Ⓓ $f_a(x)=ax \quad f'(x)=a$

② INSERT DERIVATIVES



G $f(x,y) = x+y \quad \frac{\partial f(x,y)}{\partial x} = 1 \quad \frac{\partial f(x,y)}{\partial y} = 1$

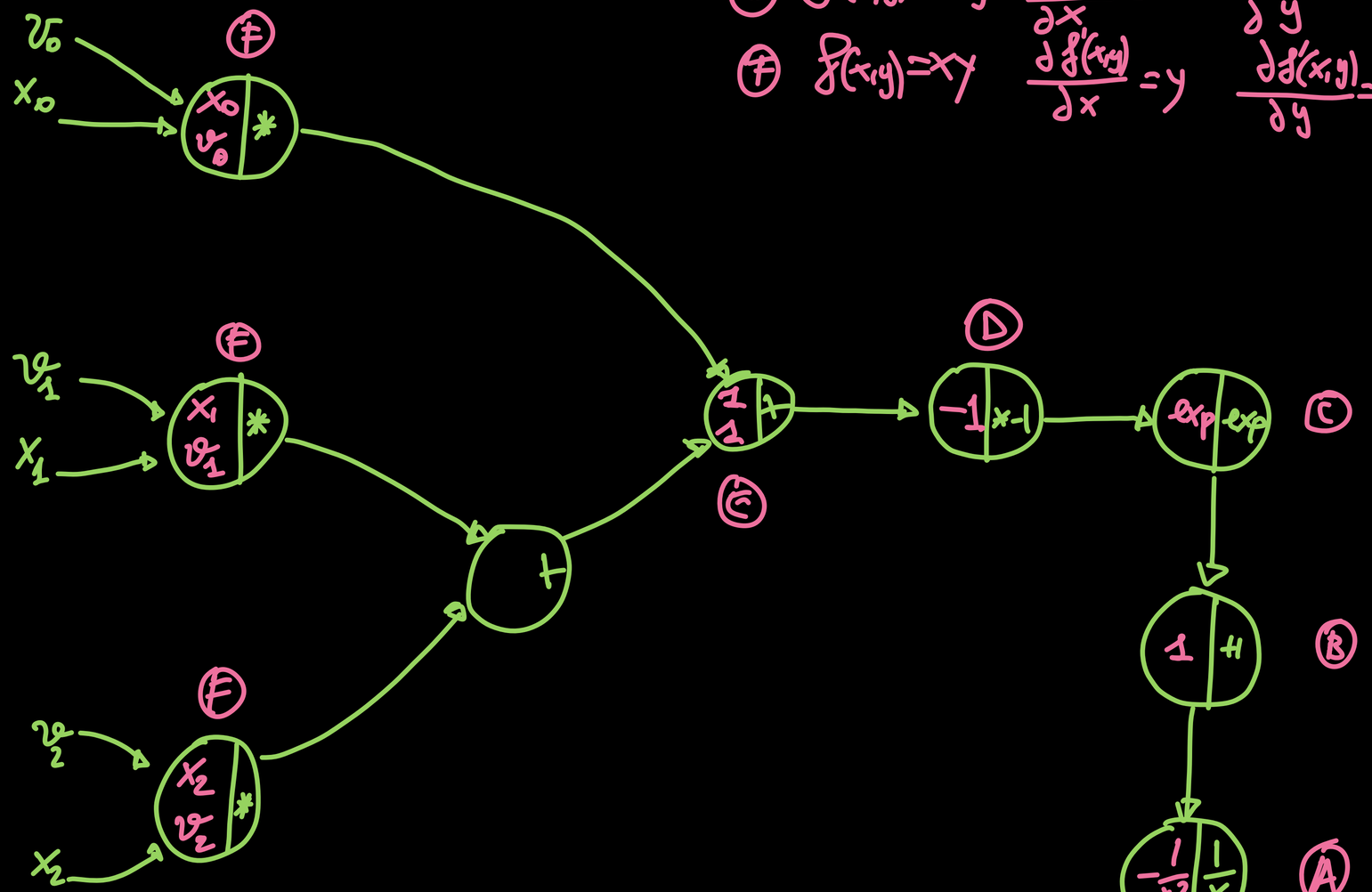
A $f(x) = \frac{1}{x} \Rightarrow f'(x) = -\frac{1}{x^2}$

B $f_c(x) = c + x \Rightarrow f'(x) = 1$

C $f(x) = e^x \Rightarrow f'(x) = e^x$

D $f_a(x) = ax \quad f'(x) = a$

② INSERT DERIVATIVES



G $f(x,y) = x+y$ $\frac{\partial f(x,y)}{\partial x} = 1$ $\frac{\partial f'(x,y)}{\partial y} = 1$

F $f(x,y) = xy$ $\frac{\partial f(x,y)}{\partial x} = y$ $\frac{\partial f'(x,y)}{\partial y} = x$

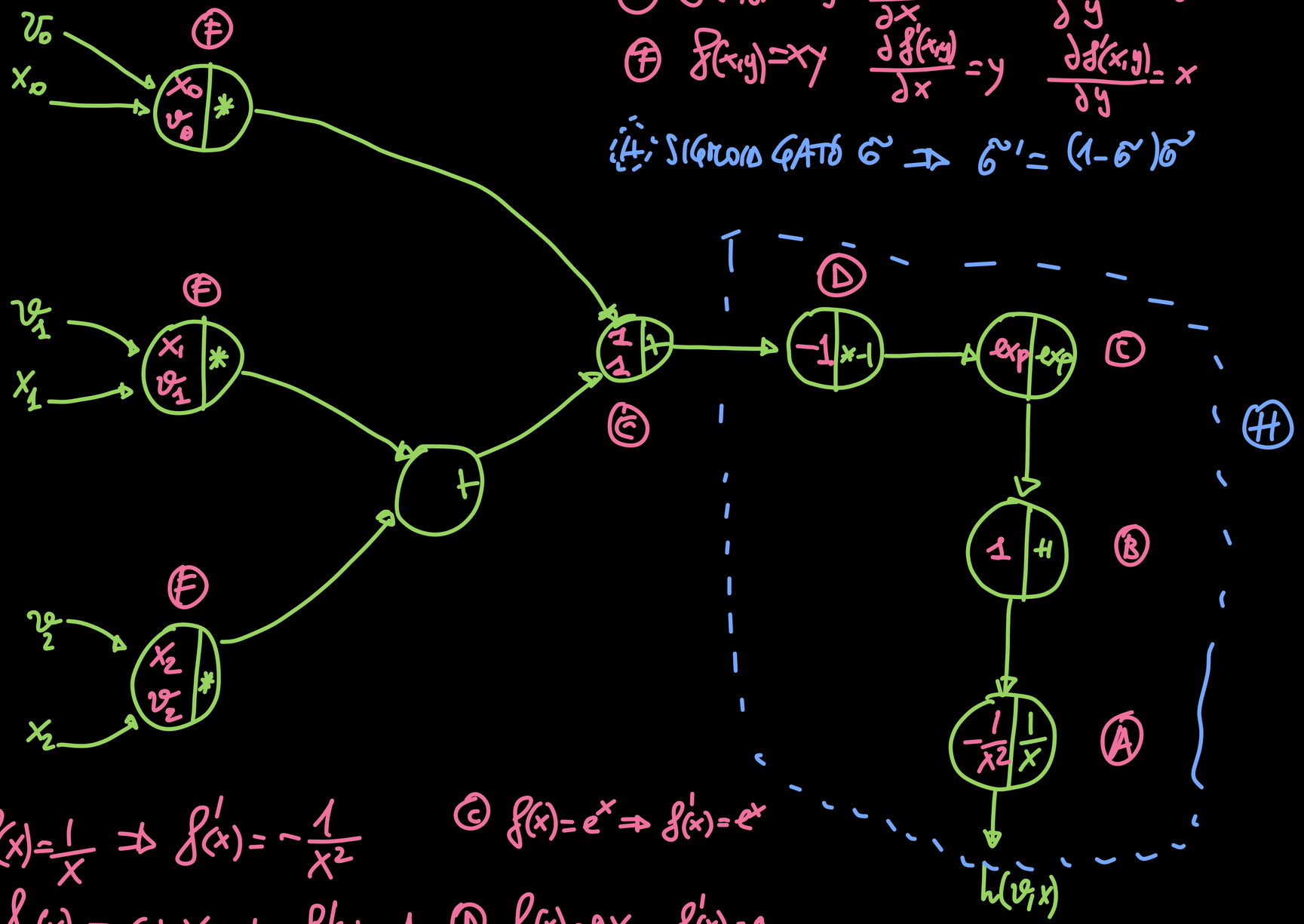
A $f(x) = \frac{1}{x} \Rightarrow f'(x) = -\frac{1}{x^2}$

B $f_c(x) = c + x \Rightarrow f'(x) = 1$

C $f(x) = e^x \Rightarrow f'(x) = e^x$

D $f_a(x) = ax$ $f'(x) = a$

② INSERT DERIVATIVES



G $f(x,y)=x+y$ $\frac{\partial f(x,y)}{\partial x}=1$ $\frac{\partial f'(x,y)}{\partial y}=1$

F $f(x,y)=xy$ $\frac{\partial f(x,y)}{\partial x}=y$ $\frac{\partial f'(x,y)}{\partial y}=x$

H: SIGMOID GATE σ ⇒ $\sigma'=(1-\sigma)\sigma$

A $f(x)=\frac{1}{x}$ ⇒ $f'(x)=-\frac{1}{x^2}$

B $f_c(x)=c+x$ ⇒ $f'(x)=1$

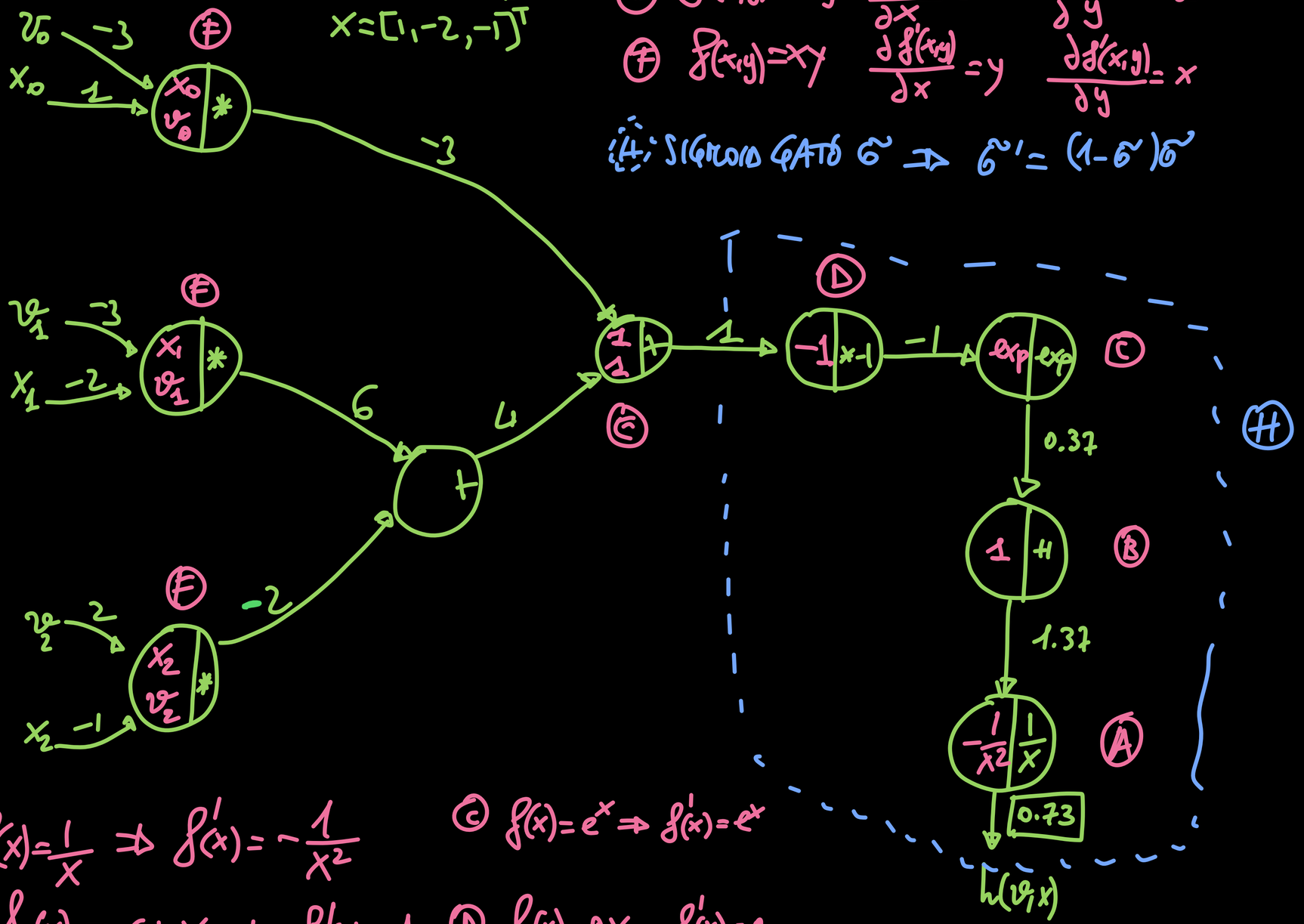
C $f(x)=e^x$ ⇒ $f'(x)=e^x$

D $f_a(x)=ax$ $f'(x)=a$

③ ADD FEEDFORWARD VALUES

$v = [-3, -3, -2]^T$

$x = [1, -2, -1]^T$



E $f(x,y) = x+y$ $\frac{\partial f(x,y)}{\partial x} = 1$ $\frac{\partial f(x,y)}{\partial y} = 1$

F $f(x,y) = xy$ $\frac{\partial f(x,y)}{\partial x} = y$ $\frac{\partial f(x,y)}{\partial y} = x$

H: SIGMOID GATE $\sigma \Rightarrow \sigma' = (1-\sigma)\sigma$

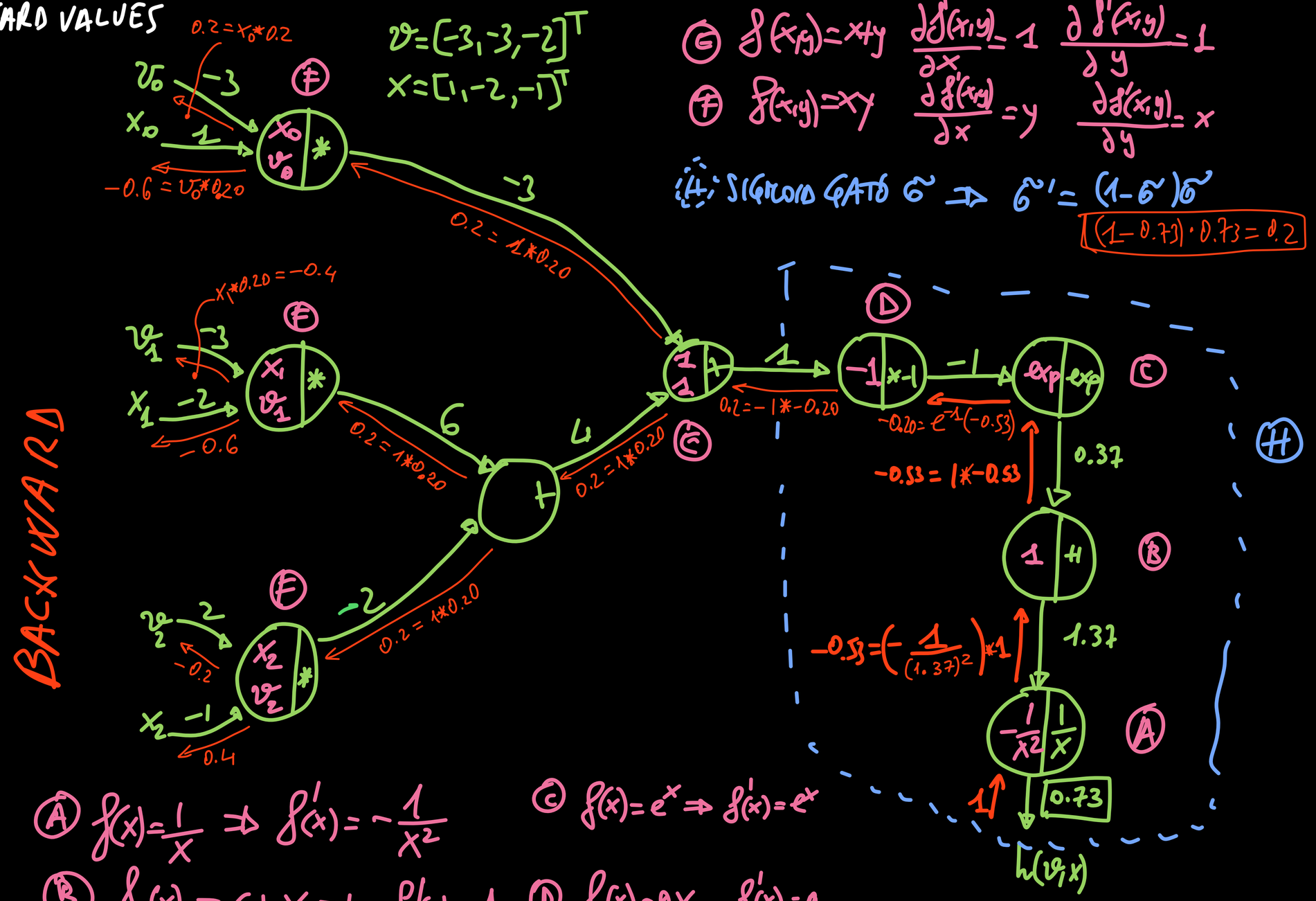
A $f(x) = \frac{1}{x} \Rightarrow f'(x) = -\frac{1}{x^2}$

B $f_c(x) = c + x \Rightarrow f'(x) = 1$

C $f(x) = e^x \Rightarrow f'(x) = e^x$

D $f_a(x) = ax$ $f'(x) = a$

④ ADD BACKWARD VALUES



⑤ $f(x,y) = x+y$ $\frac{\partial f(x,y)}{\partial x} = 1$ $\frac{\partial f'(x,y)}{\partial y} = 1$

⑥ $f(x,y) = xy$ $\frac{\partial f(x,y)}{\partial x} = y$ $\frac{\partial f'(x,y)}{\partial y} = x$

④: SIGMOID GATE $\sigma \Rightarrow \sigma' = (1-\sigma)\sigma$

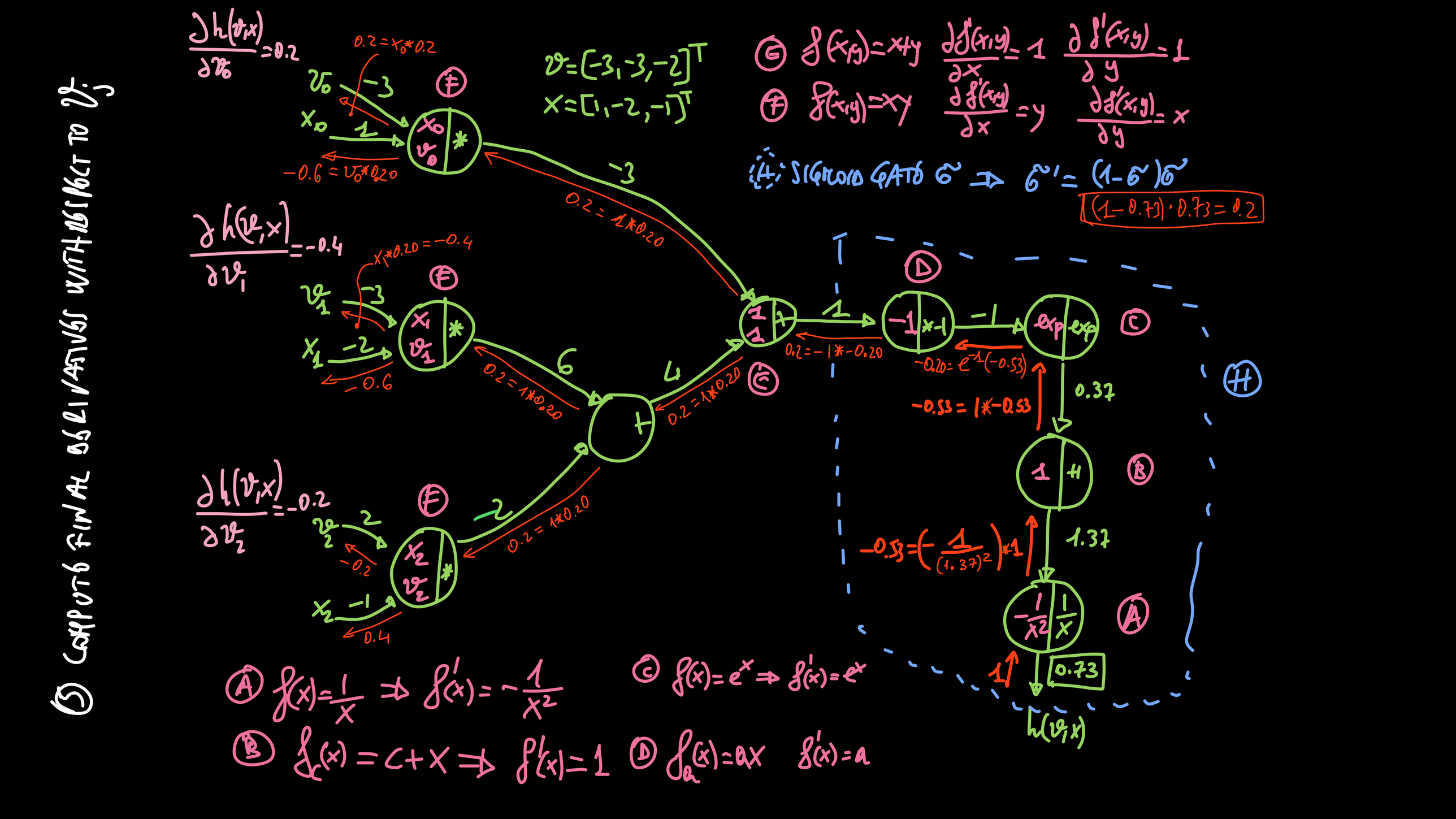
$(1-0.73) \cdot 0.73 = 0.2$

Ⓐ $f(x) = \frac{1}{x} \Rightarrow f'(x) = -\frac{1}{x^2}$

Ⓑ $f_c(x) = c + x \Rightarrow f'(x) = 1$

Ⓒ $f(x) = e^x \Rightarrow f'(x) = e^x$

Ⓓ $f_a(x) = ax$ $f'(x) = a$


5) COMPUTE FINAL DERIVATIVE WITH RESPECT TO v_j
$\frac{\partial h(v,x)}{\partial v_0} = 0.2$
$\frac{\partial h(v,x)}{\partial v_1} = -0.4$
$\frac{\partial h(v,x)}{\partial v_2} = -0.2$
$v = [-3,-3,-2]^T$
$x = [1,-2,-1]^T$
G $f(x,y) = x+y$ $\frac{\partial f(x,y)}{\partial x} = 1$ $\frac{\partial f(x,y)}{\partial y} = 1$
F $f(x,y) = xy$ $\frac{\partial f(x,y)}{\partial x} = y$ $\frac{\partial f(x,y)}{\partial y} = x$
H: SIGMOID GATE σ ⇒ $\sigma' = (1-\sigma)\sigma$
$(1-0.73)\cdot 0.73 = 0.2$
A $f(x) = \frac{1}{x} \Rightarrow f'(x) = -\frac{1}{x^2}$
B $f_c(x) = c+x \Rightarrow f'(x) = 1$
C $f(x) = e^x \Rightarrow f'(x) = e^x$
D $f_a(x) = ax$ $f'(x) = a$
$h(v,x)$

HOMEWORK

Applicare l'algoritmo della backpropagation per derivare (utilizzando e disegnando grafo computazionale) la funzione riportata sotto rispetto ai parametri θ_i all'iterazione *t*-esima dell'algoritmo della discesa del gradiente considerando $x = [1,2,1]^T$ come input e $\theta = [1,1,1]^T$ al passo *t-1*:

$$f(x_0, x_1, x_2) = e^{-(\theta_0 * x_0 + \theta_1 * x_1^2 + \theta_2 * x_2^3)}$$

Indicare quindi i valori di $\frac{\partial f(x_0,x_1,x_2)}{\partial \theta_i}$ per i=0,1,2. Calcolare quindi il nuovo vettore θ dei parametri una volta applicata la discesa del gradiente con learning rate α=2.

Initializing Neural Networks

Further Readings

Go to the link for “Tips and Tricks” and further readings