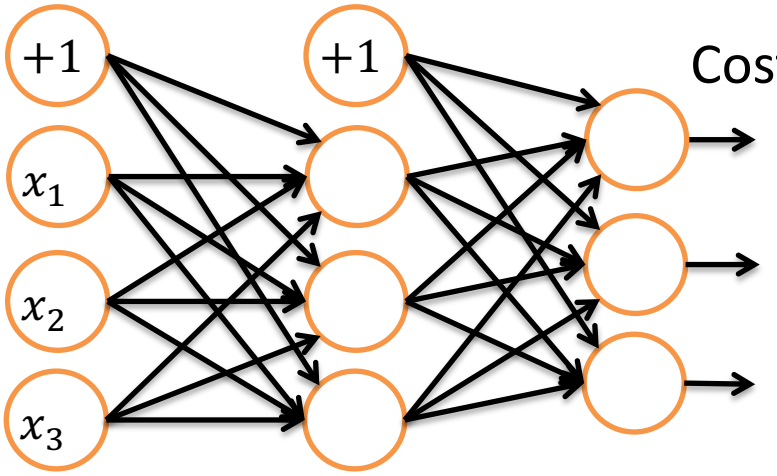
Backpropagation Algorithm

Multi-Class Classification



Cost Function

$y^{(i)} \in \mathbb{R}^k$ $\quad y = \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \ldots$

#classi: $\mathrm{K} = 3$

One-hot vector for categorical classification

Cost Function for Binary and Multi-Label

$$\mathrm{J}(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[{y_k}^{(i)}\log\left(\left(h_\theta\left(x^{(i)}\right)\right)_k\right) + \left(1-{y_k}^{(i)}\right)\log\left(1-\left(h_\theta\left(x^{(i)}\right)\right)_k\right)\right]$$

Cost Function for Categorical

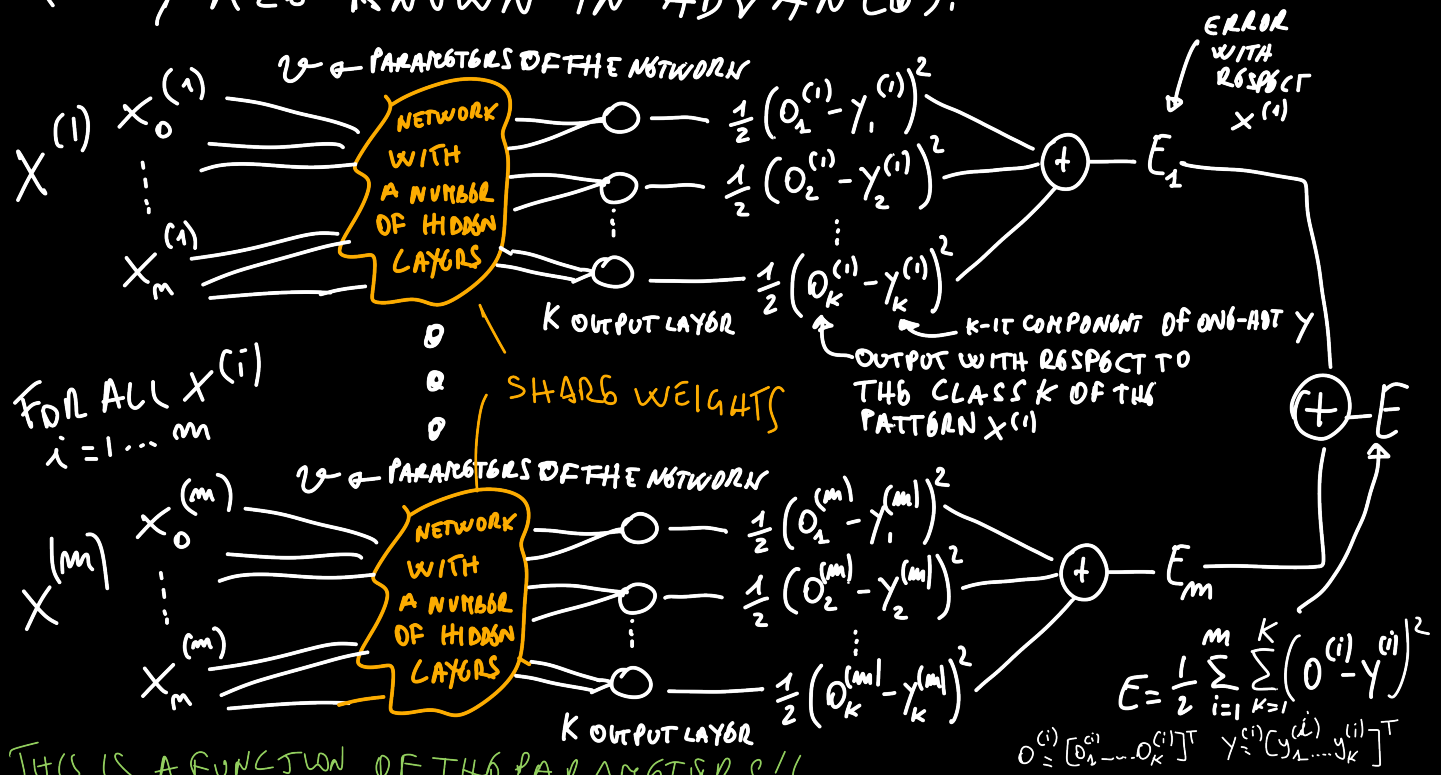
$$\mathrm{J}(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[{y_k}^{(i)}\log\left(\left(h_\theta\left(x^{(i)}\right)\right)_k\right)\right]$$

Goal is the minimization of the cost function → Gradient Descent

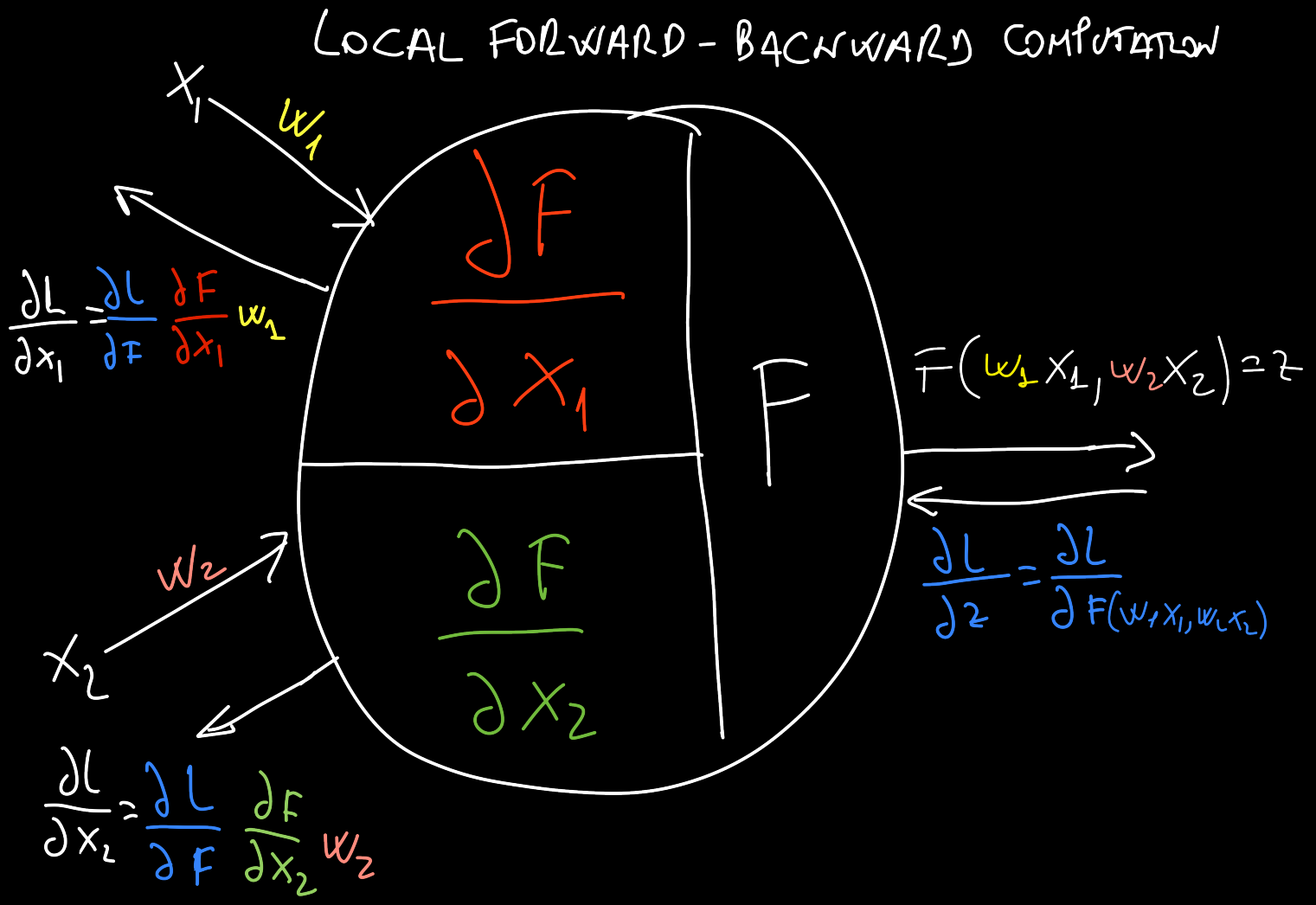
$\min\{J(\theta)\}$ → need to compute $\frac{d\mathrm{J}(\theta)}{d{\theta_{j,i}}^{(l)}}$ $\forall i, j, l$

Extended Network

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



Computational Graphs

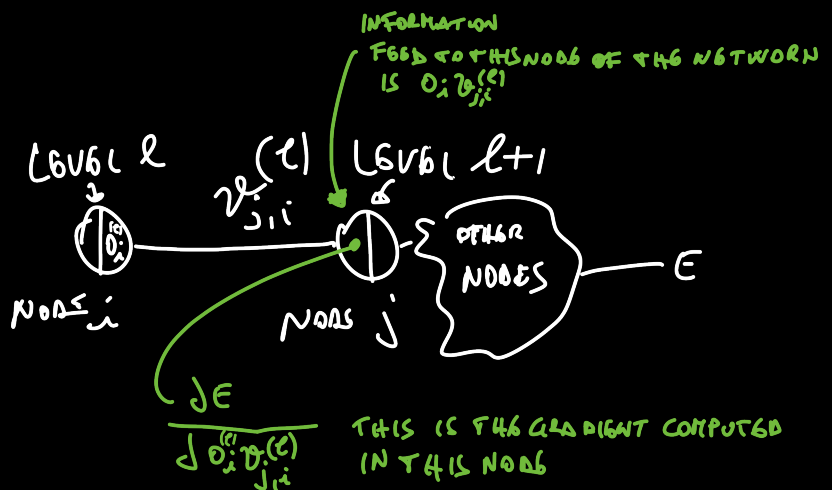


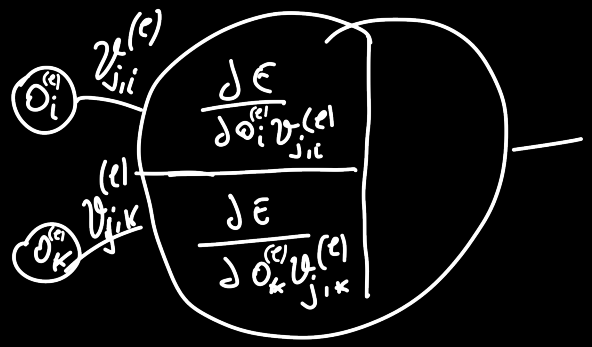
Computational Graphs
Partial Derivative

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





Computational Graphs After Forward Pass

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

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

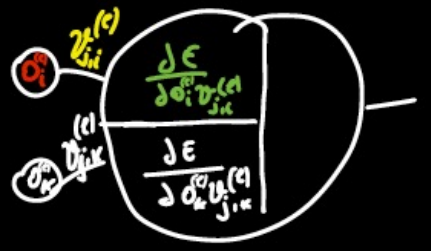
$$\frac{\partial E}{\partial v_{ji}^{(e)}} = \frac{\partial o_i^{(e)} v_{ji}^{(e)}}{\partial v_{ji}^{(e)}} \frac{\partial E}{\partial o_i^{(e)} v_{ji}^{(e)}} = o_i^{(e)} \boxed{\frac{\partial E}{\partial o_i^{(e)} v_{ji}^{(e)}}}$$

OUR DESIDERATA

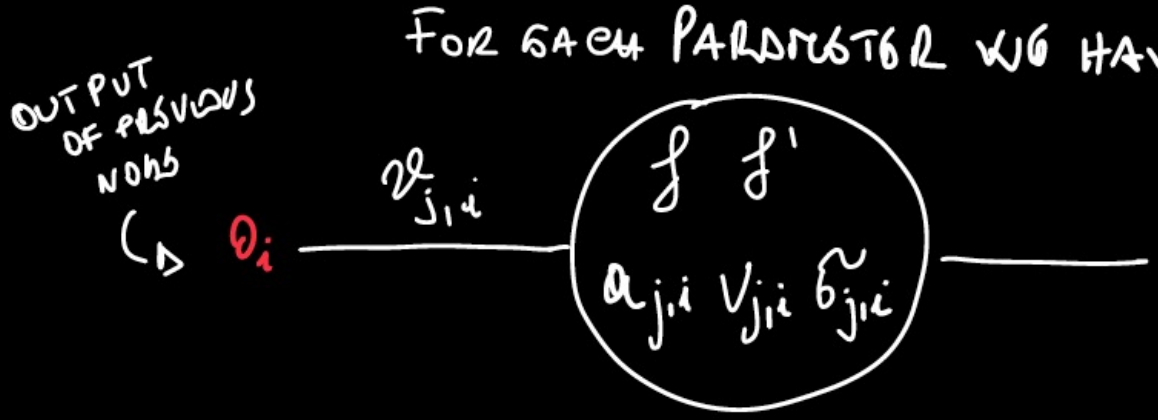
FOR THE CHAIN RULE

BACKPROPAGATION ERROR AT NODE j

(USUALLY DENOTED WITH $\delta_{ji}^{(e)}$)



Computational Graphs
Nodes



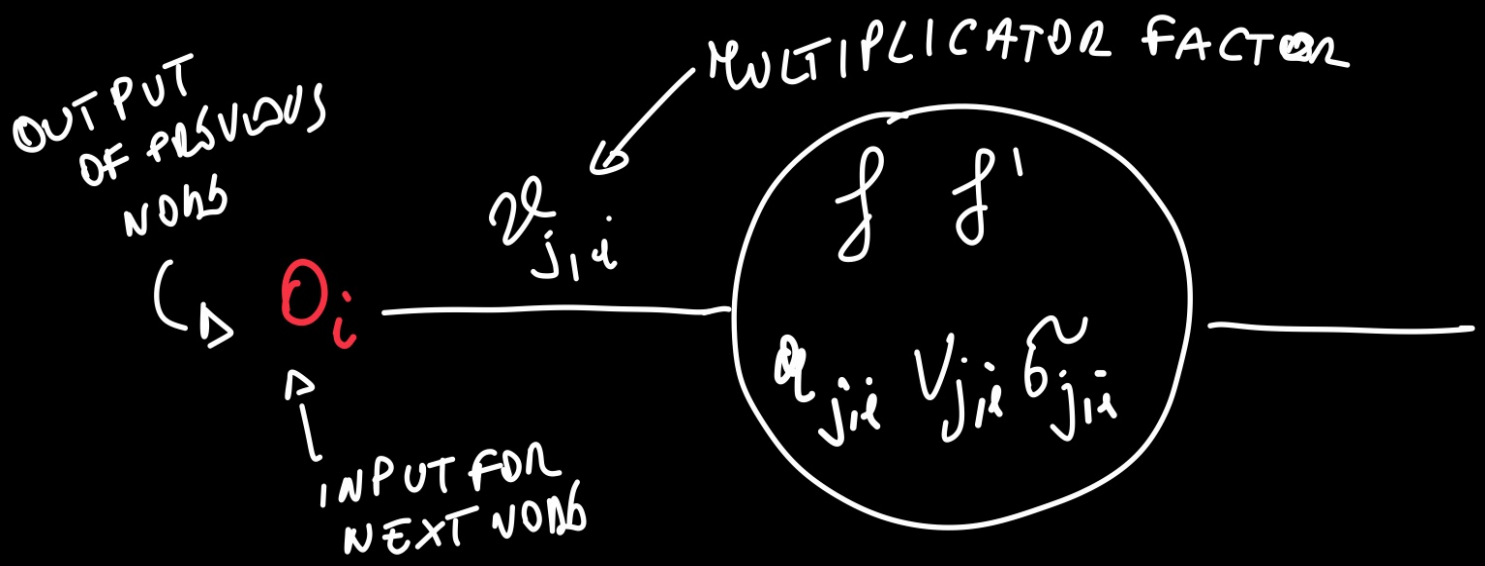
f: PRIMITIVE FUNCTION TO BE COMPUTED ON THE INPUT $O_i v_{j,i}$

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 FEEDFORWARD 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

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

THIS CAN BE DONE FOR EACH PARAMETER OF THE NET

Last Layer

SOFTMAX: $\hat{y}_i = \frac{e^{z_i}}{\Sigma_K} \quad \forall\, i = 1 \dots K$ WHERE $\Sigma_K = \sum_{K=1}^{K} e^{z_K}$

DERIVATIVE OF SOFTMAX

TO USE SOFTMAX IN NEURAL NET, WE NEED TO COMPUTE ITS DERIV-ITVE.

WE NEED $\frac{\partial \hat{y}_i}{\partial z_j}$ OF THE OUTPUT OF THE VECTOR $\hat{y}$ WITH RESPECT TO THE INPUT VECTOR z:

IF $i = j \Rightarrow \frac{\partial \hat{y}_i}{\partial z_i} = \frac{\partial \frac{e^{z_i}}{\Sigma_K}}{\partial z_i} = \frac{e^{z_i}\Sigma_K - e^{z_j}e^{z_i}}{\Sigma_K^2} = \frac{e^{z_i}}{\Sigma_K}\frac{\Sigma_K - e^{z_j}}{\Sigma_K} = \hat{y}_i(1-\hat{y}_i)$

IF $i \neq j \Rightarrow \frac{\partial \hat{y}_i}{\partial z_j} = \frac{\partial \frac{e^{z_i}}{\Sigma_K}}{\partial z_j} = \frac{0 - e^{z_i}e^{z_j}}{\Sigma_K^2} = -\frac{e^{z_i}}{\Sigma_K}\frac{e^{z_j}}{\Sigma_K} = -\hat{y}_i\hat{y}_j$

Loss Function

$\hat{y}$ = SOFTMAX OUTPUT

y = LABEL OF PATTERN x

CROSS-ENTROPY: $-\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\hat{y}_k^{(i)}\right) = L$

DERIVATIVE OF CROSS-ENTROPY FOR THE SOFTMAX FUNCTION

$$\frac{\partial L}{\partial z_i} = -\sum_{k=1}^{K} \frac{\partial y_k \log(\hat{y}_k)}{\partial z_i} = -\sum_{k=1}^{K} y_k \frac{\partial \log(\hat{y}_k)}{\partial z_i} = -\sum_{k=1}^{K} y_k \frac{1}{\hat{y}_k}\frac{\partial \hat{y}_k}{\partial z_i}$$

$$= -\frac{y_k}{\hat{y}_k}\frac{\partial \hat{y}_k}{\partial z_k} - \sum_{j\neq k}^{K} \frac{y_j}{\hat{y}_j}\frac{\partial \hat{y}_j}{\partial z_k} = -\frac{y_k}{\hat{y}_k}\hat{y}_k(1-\hat{y}_k) - \sum_{j\neq k}^{K} \frac{y_j}{\hat{y}_j}(-\hat{y}_j\hat{y}_k)$$

SEE PREVIOUS SLIDE

$$= -y_k + y_k\hat{y}_k + \sum_{j\neq k}^{K} y_j\hat{y}_k = -y_k + \sum_{j=1}^{K} y_j\hat{y}_k = -y_k + \hat{y}_k\sum_{j=1}^{K} y_j =$$

$$= \hat{y}_k - y_k$$

Multi-Class Classification

Neural Network - Training

Training:

1) Parameters Initialization (with simmetry breaking)

2) For every sample ($x^{(i)}$, $y^{(i)}$) of the training set

   - Forward propagation of $x^{(i)}$
   - Backpropagation considering $y^{(i)}$ and derivative computation $\frac{d\mathrm{J}(\theta)}{d\theta_{j,i}^{(l)}}$

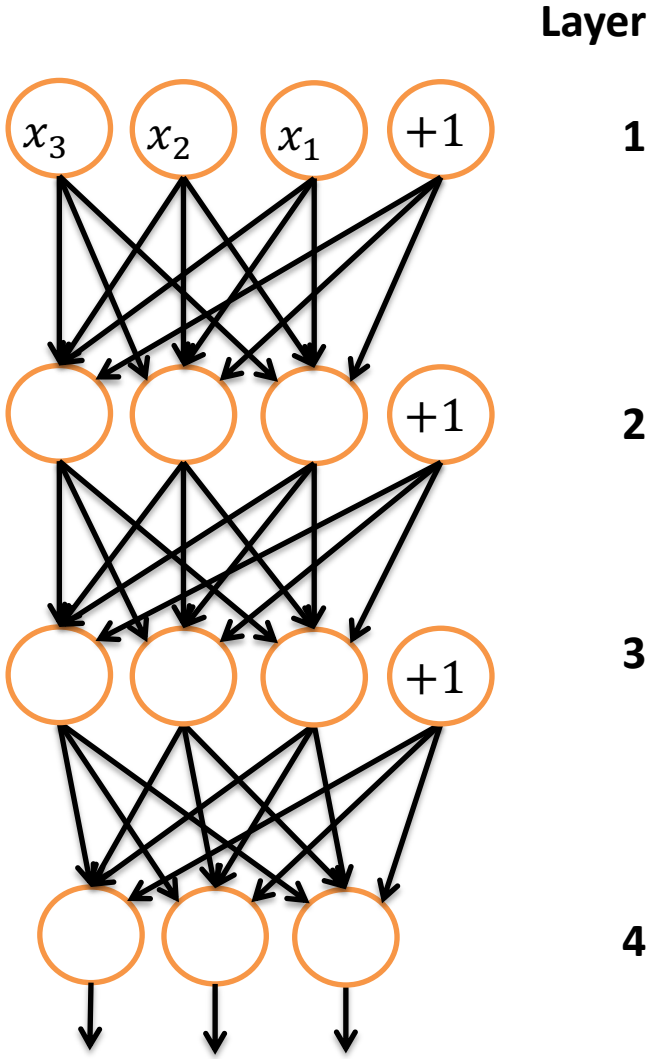
3) Parameters upgrade with Gradient Descent

Multi-Class Classification

Forward Propagation

Computing of activations, $(x^{(i)}, y^{(i)})$

$a^{(1)} = x^{(i)}$



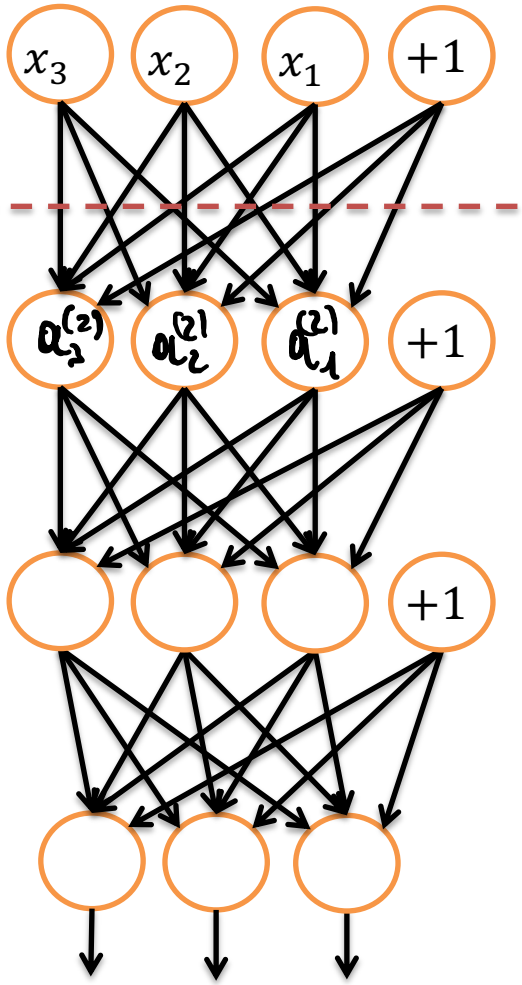
Multi-Class Classification

Forward Propagation

Calcolo delle attivazioni, $(x^{(i)}, y^{(i)})$

$a^{(1)} = x^{(i)}$

$\theta^{(1)}$



$z^{(2)} = \theta^{(1)} a^{(1)}$

$a^{(2)} = f(z^{(2)})$

ACTIVATION FUNCTION (es. SIGMOID)

$v_j^{(2)} = f(z^{(2)})$

$$\theta^{(1)} = \begin{bmatrix} \theta_{1,0}^{(1)} & \theta_{1,1}^{(1)} & \theta_{1,2}^{(1)} & \theta_{1,3}^{(1)} \\ \theta_{2,0}^{(1)} & \theta_{2,1}^{(1)} & \theta_{2,2}^{(1)} & \theta_{2,3}^{(1)} \\ \theta_{3,0}^{(1)} & \theta_{3,1}^{(1)} & \theta_{3,2}^{(1)} & \theta_{3,3}^{(1)} \end{bmatrix}$$

$$a^{(1)} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(1)} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$$

$$a^{(2)} = \begin{bmatrix} 1 \\ a_1^{(2)} \\ a_2^{(2)} \\ a_2^{(3)} \end{bmatrix}$$

$$z_i^{(1)} = \theta_{i,0}^{(1)} + \theta_{i,1}^{(1)} x_1 + \theta_{i,2}^{(1)} x_2 + \theta_{i,3}^{(1)} x_3$$

Multi-Class Classification

Forward Propagation

Calcolo delle attivazioni, $(x^{(i)}, y^{(i)})$

$a^{(1)} = x^{(i)}$

$z^{(2)} = \theta^{(1)} a^{(1)}$

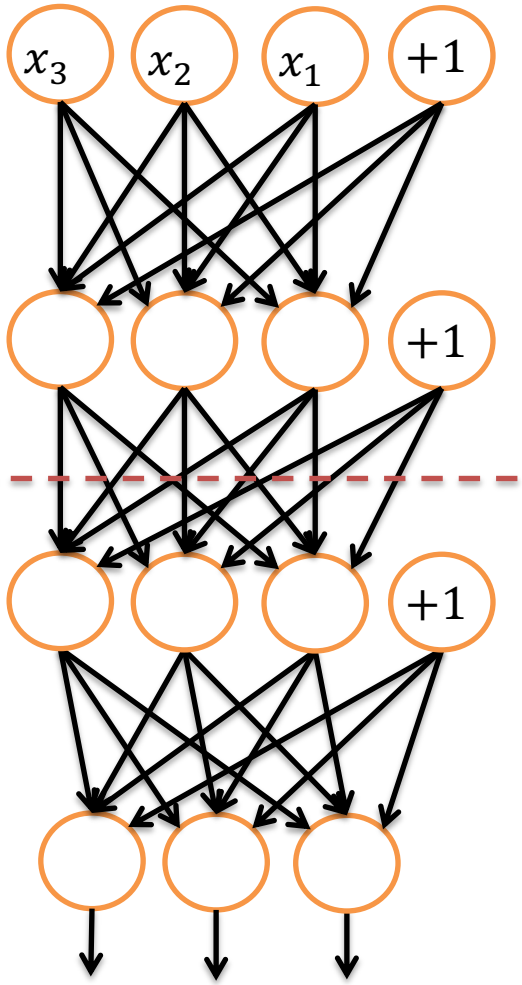
$a^{(2)} = f(z^{(2)})$

$\theta^{(2)}$

Homework

$z^{(3)} = \theta^{(2)} a^{(2)}$

$a^{(3)} = f(z^{(3)})$



Multi-Class Classification

Forward Propagation

Calcolo delle attivazioni, $(x^{(i)}, y^{(i)})$

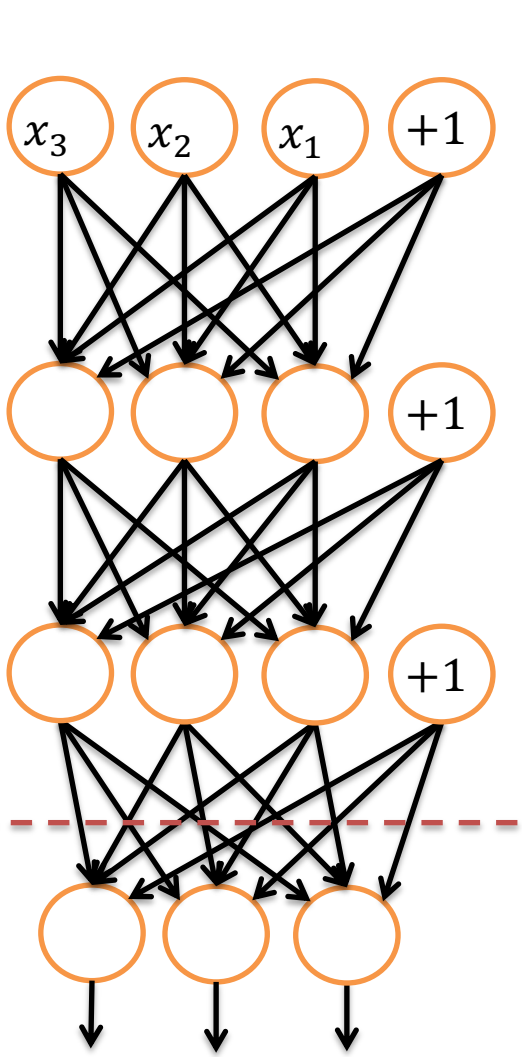
$a^{(1)} = x^{(i)}$

$z^{(2)} = \theta^{(1)} a^{(1)}$
$a^{(2)} = f(z^{(2)})$

$z^{(3)} = \theta^{(2)} a^{(2)}$
$a^{(3)} = f(z^{(3)})$

$\theta^{(3)}$

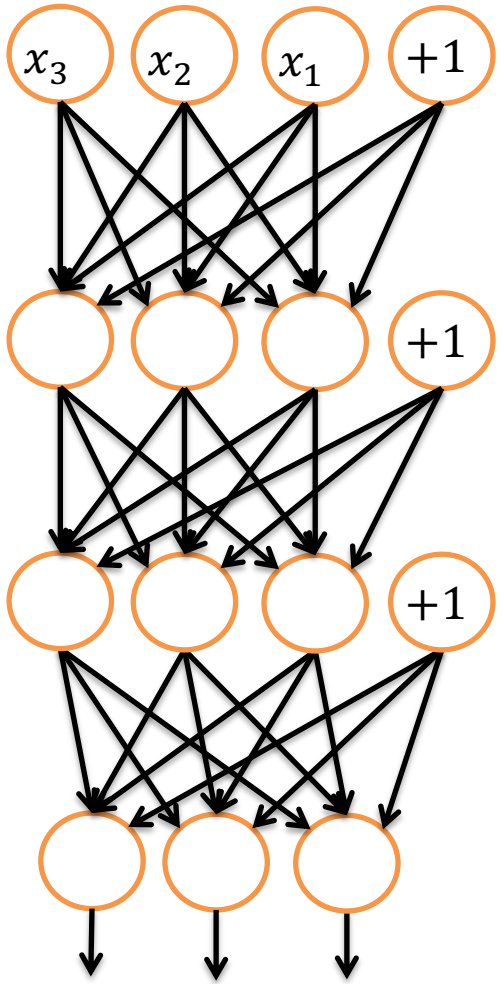
$z^{(4)} = \theta^{(3)} a^{(3)}$
$a^{(4)} = h_\theta(x^{(i)}) = f(z^{(4)})$



Multi-Class Classification

Backpropagation

Partial derivatives computation $(x^{(i)}, y^{(i)})$



$\delta^{(4)} = (a^{(4)} - y^{(i)})$

Multi-Class Classification

Backpropagation

Partial derivatives computation, $(x^{(i)}, y^{(i)})$

$$\left(\theta^{(3)}\right)^T \delta^{(4)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \quad 4\times 1$$

$$\theta_{1,1}^{(3)}\,\delta_1^{(4)} + \theta_{2,1}^{(3)}\,\delta_2^{4} + \theta_{3,1}^{(3)}\,\delta_3^{4}$$

$$\theta^{(3)} = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \quad 3\times 4$$

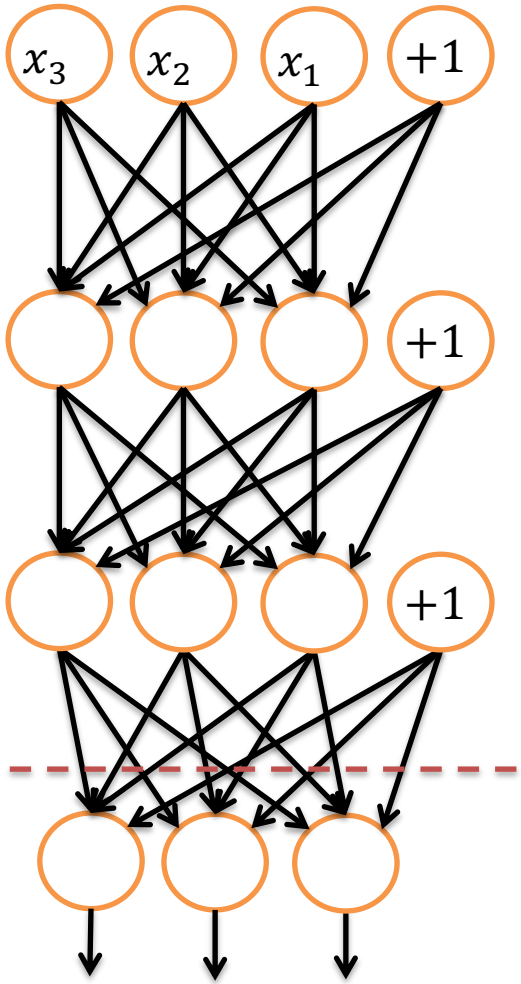
$$\left(\theta^{(3)}\right)^T = \begin{bmatrix} \theta_{1,0}^{(3)} & \theta_{2,0}^{(3)} & \theta_{3,0}^{(3)} \\ \theta_{1,1}^{(3)} & \theta_{2,1}^{(3)} & \theta_{3,1}^{(3)} \\ \theta_{1,2}^{(3)} & \theta_{2,2}^{(3)} & \theta_{3,2}^{(3)} \\ \theta_{1,3}^{(3)} & \theta_{2,3}^{(3)} & \theta_{3,3}^{(3)} \end{bmatrix} \quad \theta^{(3)}$$

3×1

$$\delta^{(3)} = \left(\theta^{(3)}\right)^T \delta^{(4)} .* f'(z^{(3)})$$

$$\delta^{(4)} = a^{(4)} - y^{(i)}$$

$$\delta^{(4)} = \begin{bmatrix} a_1^{(4)} - y_1^{(i)} \\ a_2^{(4)} - y_2^{(i)} \\ a_3^{(4)} - y_3^{(i)} \end{bmatrix} = \begin{bmatrix} \delta_1^{(4)} \\ \delta_2^{(4)} \\ \delta_3^{(4)} \end{bmatrix}$$



Multi-Class Classification

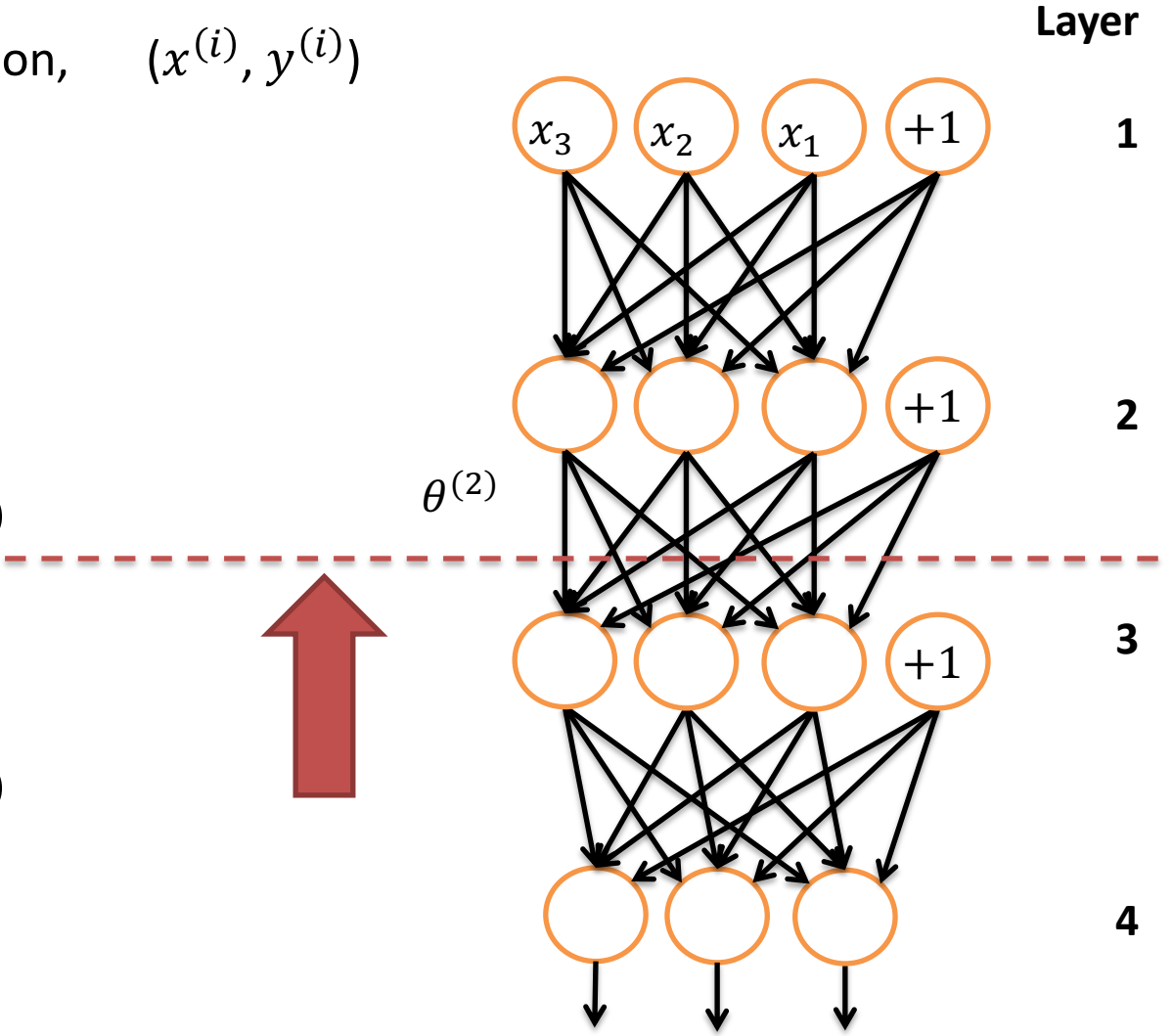
Backpropagation

Partial derivatives computation, $(x^{(i)}, y^{(i)})$

$$\delta^{(2)} = \left(\theta^{(2)}\right)^T \delta^{(3)} .* f'(z^{(2)})$$

$$\delta^{(3)} = \left(\theta^{(3)}\right)^T \delta^{(4)} .* f'(z^{(3)})$$

$$\delta^{(4)} = a^{(4)} - y^{(i)}$$



Multi-Class Classification

Backpropagation

Partial derivatives computation, $(x^{(i)}, y^{(i)})$

$$\delta^{(2)} = \left(\theta^{(2)}\right)^T \delta^{(3)} .* f'(z^{(2)}) \quad \Rightarrow \quad \frac{d\mathrm{J}(\theta)}{d\theta^{(1)}} = a^{(1)}\,\delta^{(2)}$$

$$\delta^{(3)} = \left(\theta^{(3)}\right)^T \delta^{(4)} .* f'(z^{(3)}) \quad \Rightarrow \quad \frac{d\mathrm{J}(\theta)}{d\theta^{(2)}} = a^{(2)}\,\delta^{(3)}$$

$$\delta^{(4)} = a^{(4)} - y^{(i)} \quad \Rightarrow \quad \frac{d\mathrm{J}(\theta)}{d\theta^{(3)}} = a^{(3)}\,\delta^{(4)}$$

Back Propagation

Backpropagation Algorithm:

Given a training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots (x^{(t)}, y^{(t)}), \dots (x^{(m)}, y^{(m)})\}$

$\Delta_{j,i}{}^{(l)} = 0 \quad \forall i, j, l$

$\forall t = 1, \dots, m$

$a^{(1)} = x^{(t)}$

forward propagation → $a^{(2)}, a^{(3)}, a^{(4)}, \dots, a^{(L)}$

$\delta^{(L)} = a^{(L)} - y^{(t)}$

compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$

compute $\Delta_{j,i}{}^{(l)} = \Delta_{j,i}{}^{(l)} + a_i{}^{(l)} \delta_{j,i}{}^{(l+1)}, l = 1, \dots L-1$

$\frac{dJ(\theta)}{d\theta_{j,i}{}^{(l)}} = a_i{}^{(l)} \delta_{j,i}{}^{(l+1)}$

$D_{j,i}{}^{(l)} = \frac{1}{m} \Delta_{j,i}{}^{(l)}$ if j=0

$D_{j,i}{}^{(l)} = \frac{1}{m} \left(\Delta_{j,i}{}^{(l)} + \lambda \theta_{j,i}{}^{(l)}\right)$ Otherwise

Regularization

Can be used to upgrade the parameters in the Gradient