

$n_inputs + 1$
 w_jli → è il bias del layer dopo
 io cioè la moltiplica di
 pesi come:

Bias
NEL
FORWARD

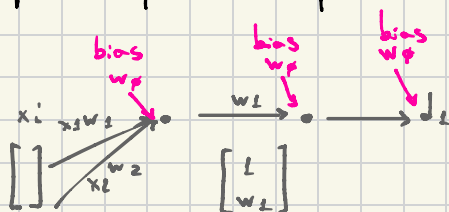
$w_jli \Rightarrow n_inputs + bias \times n_hidden$
 Quindi avrà sempre un valore in più.

Va riconsiderato il **Bias**, in tutti gli hidden layer

tipo $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}^{2 \times 3} \Rightarrow + bias = 3 \times 3$

w_{jl}
 w_jli
 $\begin{bmatrix} 1 & 1 & 1 \\ w_{11} & w_{21} & w_{31} \\ w_{12} & w_{22} & w_{32} \end{bmatrix}$

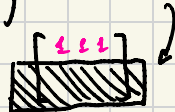
poi moltiplico in forward il vettore $x-i$



, nel net cioè il bias =

$$\left(\sum_i w_{ji} + x_i \right) + w_{j0}$$

quindi un qui e poi il
 sogli domanda bias



allora ho tutte le matrici dei pesi come:

w_jli w_jli w_kjl
 $\begin{bmatrix} [1 \dots 1] \\ \diagdown \end{bmatrix}$ $\begin{bmatrix} [1 \dots 1] \\ \diagdown \end{bmatrix}$ $\begin{bmatrix} [1 \dots 1] \\ \diagdown \end{bmatrix}$

che sono considerate nel forward e nell'aggiornam. poi,
 perciò vanno modificate le funzioni

forward

$$\begin{bmatrix} 1 & \dots & 1 \\ \vdots & & \vdots \\ \vdots & & \vdots \end{bmatrix}^{k_{unit}} \Rightarrow \text{np.dot}(\text{vettore } x_i, w_{ji}[1:, j_{unit}]) + w_{ji}[0][j_{unit}]$$

che mi restituisce il vettore risultato relativo al layer.
Dopo questa fase ho il risultato della rete che confronta trovando l'errore e propagandolo all'indietro, come si comporta il bias in questo caso?

Bias
nella
BACKPROP.

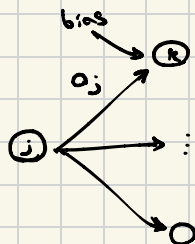
δ_k
Questo è semplice.
il vettore δk_{unit}
($a_k - o_k$)

procedimento simile
per δ_j
tranne per
una cosa *

$$\Rightarrow w_{kj}^{new} = w_{kj} + \eta \cdot \delta_k \cdot o_j$$

Però w_{kj} ha come prima
riga il bias = il primo elemento
del vettore w_{kj} , fissando k , non
ha un collegamento con un'unità nel layer precedente.
Quindi quando calcolo Δw_{kj} , che contiene il bias dell'output layer
ho: $\Delta_p w_{kj} = \delta_k \cdot o_j$ ma per $\Delta_p w_{k\phi}$?

$$\begin{bmatrix} 1 & \dots & 1 \\ \vdots & & \vdots \\ \vdots & & \vdots \end{bmatrix}^{k_{unit}}$$



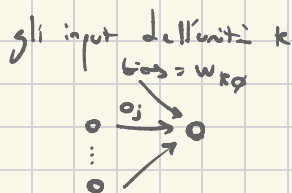
$$o_j = f_j(\text{net}_j)$$

vediamo la formula:

Il gradiente rispetto a $\Delta_P w_{kp}$ =

$$- \frac{\partial E_P}{\partial \text{net}_k} \cdot \frac{\partial \text{net}_k}{\partial w_{kp}} = \dots$$

$$\frac{\partial (\sum_j w_{kj} o_j) + w_{kp}}{\partial w_{kp}} = 1$$



BIAS NELL'UPDATE WEIGHTS

Quindi nel caso δ_k ho:

non bias $\rightarrow \Delta_P w_{kj} = \delta_k \cdot o_j \Rightarrow w_{kj}^{\text{new}} = w_{kj}^{\text{old}} + \eta \cdot \delta_k \cdot o_j$

bias $\rightarrow \Delta_P w_{kp} = \delta_k \cdot 1 \Rightarrow w_{kp}^{\text{new}} = w_{kp}^{\text{old}} + \eta \cdot \delta_k \cdot 1$

e sempre per similitudine con δ_j ho:

non bias $\rightarrow \dots \Rightarrow w_{ji}^{\text{new}} = w_{ji}^{\text{old}} + \eta \cdot \delta_j \cdot o_i$

bias $\rightarrow \dots \Rightarrow w_{jp}^{\text{new}} = w_{jp}^{\text{old}} + \eta \cdot \delta_j \cdot 1$

*

ATTENZIONE

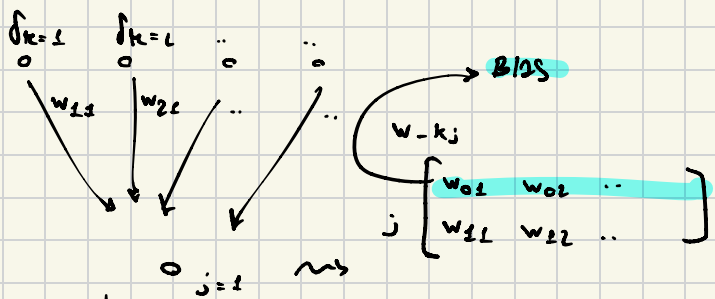
Nel calcolo del δ_j ho:

$$\delta_j = \left(\sum_{k=2}^K \delta_k \cdot w_{kj} \right) \cdot f'_j(\text{net}_j)$$

non c'è il bias

In questo pattern ho:

c'è il bias



e qui il Bias non è incluso, quindi nel calcolo di δ_j ho bisogno di sottrarre la prima riga e poi prodotto scalare tra δ_k e $w_{kj}[2, :]$

Quindi il δ_j si calcola:

$$n_units = x_j.size$$

$$d_j = np.zeros(n_units)$$

Ad unità j ho bisogno di calcolare il suo δ_j :

$$\delta_j = \left[\sum_{k=1}^K \delta_k \cdot w_{kj} \right] \left[f'_j(net_j) \right]$$

for unit in n_units:

$$(1) \rightarrow np.dot(x_i, w_ji[1:unit]) + w_j1[0][unit]$$

$$(2) \rightarrow np.dot(\delta_k, w_kj[unit+1, :])$$

$$\delta_j[unit] = (1) + f'_{act}((2))$$