LOGISTIC REGRESSION

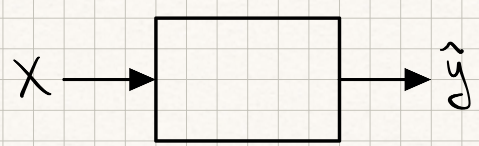
Fin ora abbiamo usato l'MLP per risolvere problemi di regressione lineare. Possiamo usarlo anche per risolvere problemi di Regressione Logistica.

La Regressione Logistica cerca di risolvere un problema di classificazione. L'obiettivo è calcolare la probabilità:

$$P(y=1 \mid x)$$

Ovvero la probabilità che, per l'ingresso X appartenga alla classe positiva ($y=1$). Per risolvere questo compito devo allenare i pesi $w_1 \dots w_m$ associati alle feature d'ingresso così da decidere se "I dati stanno sotto o sopra l'equazione $w_1x_1 + \dots + w_mx_m$", e quindi associarli a una delle due classi (Positiva o Negativa).

Prendiamo un ingresso $X \in \mathbb{R}^R$, dato a un modello che da in uscita $\hat{y} \in [0, 1]$



Il nostro modello stima la probabilità di X di appartenere alla classe C:

$$\hat{y}(x) = P(X \in C)$$

Usando come funzione di attivazione la sigmoide:

$$\hat{y}(x, w) = \sigma(w^T \cdot ext(x))$$

Threshold

Per decidere l'appartenenza a una classe, data la probabilità di appartenenza, bisogna settare una soglia oltre il quale si decide di associare l'ingresso alla classe positiva.

BINARY CROSS-ENTROPY

Per allenare un modello di tipo logistic regression, si usa come Loss Function la BCE:

$$\mathcal{L}^{BCE} := -y \log \hat{y} - (1-y) \log (1-\hat{y})$$

Questa misura la dissimilarità fra l'uscita predetta e quella attuale.

Questa è una loss convessa, per cui possiamo raggiungere l'ottimo:

$$\frac{\partial \mathcal{L}^{BCE}}{\partial w} = \frac{\partial \mathcal{L}^{BCE}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w}$$

Dove:

1) $$\boxed{\frac{\partial \mathcal{L}^{BCE}}{\partial \hat{y}}} = -y \cdot \frac{1}{\hat{y}} - (1-y) \frac{1}{1-\hat{y}} \cdot (-1) = \boxed{-\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}}$$

2) $$\boxed{\frac{\partial \hat{y}}{\partial w}} = \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial w} = \underbrace{\sigma(z)(1-\sigma(z))}_{\sigma(z)=\hat{y}} \cdot \underbrace{\frac{\partial w^T x}{\partial w}}_{x} = \boxed{\hat{y}(1-\hat{y})\, x}$$

$$-\frac{y}{\hat{y}}+\frac{1-y}{1-\hat{y}}=\frac{-y(1-\hat{y})+\hat{y}(1-y)}{\hat{y}(1-\hat{y})}=\frac{-y+\cancel{\hat{y}y}+\hat{y}-\cancel{y\hat{y}}}{\hat{y}(1-\hat{y})}=\frac{\hat{y}-y}{\hat{y}(1-\hat{y})}$$

Calcolando ora il gradiente della loss:

$$\frac{\partial \mathcal{L}^{BCE}}{\partial W}=\frac{\hat{y}-y}{\cancel{\hat{y}(1-\hat{y})}}\,\cancel{\hat{y}(1-\hat{y})}\,x=(\hat{y}-y)\,x$$

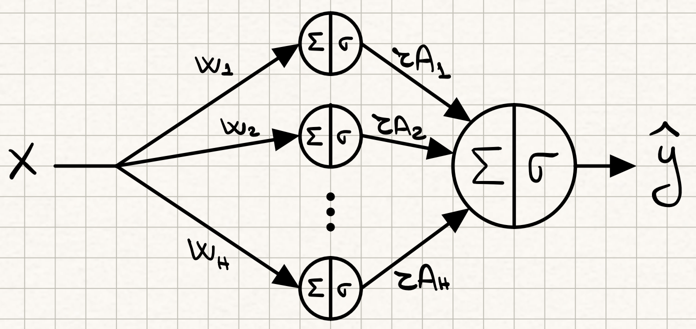
Se usiamo un mini-batch, x diventa una matrice $X \in \mathbb{R}^{B\times(D+1)}$:

$$\frac{\partial \mathcal{L}^{BCE}}{\partial W}=X^{T}(\hat{y}-y)$$

Vettori colonna

PROBLEMA DI CLASSIFICAZIONE MULTICLASSE

L'idea è sempre dare una classe all'ingresso X:



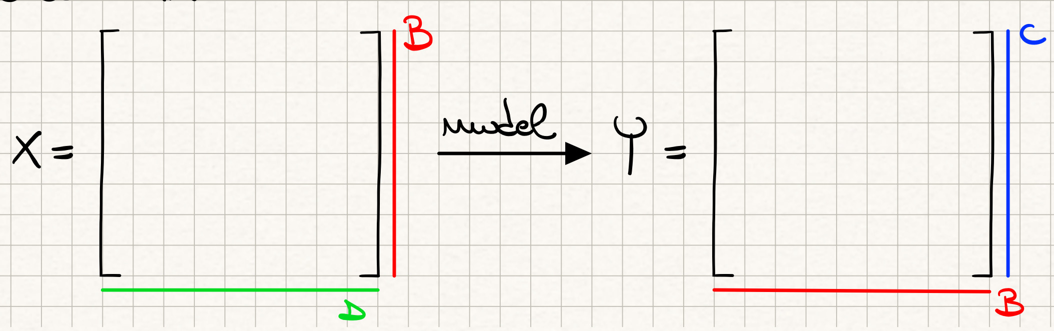
Dove $\hat{y}$ è la probabilità dell'ingresso di appartenere a una certa classe.

Nei problemi multiclasse abbiamo tanti neuroni nel layer di output quante sono le classi target.
Abbiamo quindi un vettore in uscita che ha i valori di probabilità di appartenenza a tutte le classi di training:

$$\hat{y} = \begin{bmatrix} 0.1 \\ 0.7 \\ 0.9 \\ 0.3 \end{bmatrix}$$

Per esempio se abbiamo un'immagine contenente più oggetti, probabilmente conterrà oggetti appartenenti alla classe 1 e 2 (0.7 e 0.9 di probabilità)

Supponendo un ingresso alla rete $X \in \mathbb{R}^{B \times D}$ avremo una uscita $Y \in \mathbb{R}^{C \times B}$



FORWARD PROPAGATION

$$rA_\varnothing = X^T \qquad \in \mathbb{R}^{D\times B}$$

$$A_\varnothing = ext(rA_\varnothing) \qquad \in \mathbb{R}^{(D+1)\times B}$$

$$rZ_1 = W_1 \cdot A_\varnothing \qquad : W_1 \in \mathbb{R}^{H\times(D+1)} \rightarrow rZ_1 \in \mathbb{R}^{H\times B}$$

$$rA_1 = \sigma(rZ_1)$$

$$A_1 = ext(rA_1) \qquad : \in \mathbb{R}^{(H+1)\times B}$$

$$rZ_2 = W_2 \cdot A_1 \qquad : W_2 \in \mathbb{R}^{C\times(H+1)} \rightarrow rZ_2 \in \mathbb{R}^{C\times B}$$

$$rA_2 = \sigma(rZ_2)$$

$$Y = rA_2 \qquad : Y \in \mathbb{R}^{C\times B}$$

LAB_05 > MLP_BCESIG_forward.m

```
% function [A0,rZ1,A1,rZ2,rA2] = MLP_BCESIG_forward(X, W1, W2)
function [rA2,A1,A0,rZ1] = MLP_BCESIG_forward(X, W1, W2)
% Compute the feedforward step

    rA0 = X';                  % rA0 is the "reduced A0" and it coincides with the transpose of the tall input matrix (the latter is BxD in size)
    A0  = MLP_extend(rA0);     % A0  = Ext(rA0).     It is the "extended" version of rA0, obtained by it by adding a row of ones as its new first row
    rZ1 = W1*A0;               % rZ1 = \sum(W1,A0).  It is the pre-activation at layer 1 (the hidden one)
    rA1 = MLP_sigmoid(rZ1);    % rA1 = \sigma(rZ1).  It is the output of the first layer (the hidden one)
    A1  = MLP_extend(rA1);     % A1  = Ext(rA1).     It is the extended version of rA1
    rZ2 = W2*A1;               % rZ2 = \sum(W2,A1).  It is the pre-activation at layer 2 (the output one)
    rA2 = MLP_sigmoid(rZ2);    % rA2 is the output of the second layer (the output one). rA2 is a column vector if B=1, otherwise it is a matrix

end
```

MATRICI DEI PESI

Per quando riguarda W_1 e W_2

$$W_1 = \begin{bmatrix} w_1^1 \\ \vdots \\ w_H^1 \end{bmatrix} \quad \text{dove } \forall\, w_1^i \in \mathbb{R}^{D+1}$$

$$W_2 = \begin{bmatrix} w_1^2 \\ \vdots \\ w_C^2 \end{bmatrix} \quad \text{dove } \forall\, w_2^i \in \mathbb{R}^{H+1}$$

LOSS FUNCTION

Abbiamo visto che per problemi con classe binaria si usa la Binary Cross-Entropy:

$$\mathcal{L}^{BCE} = -y \log \hat{y} - (1-y) \log (1-\hat{y})$$

Per problemi che hanno $C > 2$ usiamo:

$$\mathcal{L}^{TOT} = \sum_{\forall c} \mathcal{L}_c^{BCE}$$

la sommatoria della BCE di ogni classe. Ogni neurone di uscita rappresenta una classe:

$$\mathcal{L}_c^{BCE} = -y_c \log \hat{y}_c - (1-y_c) \log (1-\hat{y}_c)$$

Alleniamo ogni neurone indipendentemente e cerchiamo di minimizzare la somma di tutte le $\mathcal{L}_c^{BCE}$. Per farlo facciamo la backpropagation:

$$\frac{\partial \mathcal{L}}{\partial W_1} \quad , \quad \frac{\partial \mathcal{L}}{\partial W_2}$$

1) Derivata su W_2

$$\frac{\partial \mathcal{L}}{\partial W_2} = \underbrace{\frac{\partial \mathcal{L}}{\partial \hat{y}}} \underbrace{\frac{\partial \hat{y}}{\partial z_2}} \underbrace{\frac{\partial z_2}{\partial W_2}}$$

Possiamo calcolare ora $\partial \mathcal{L}/\partial z_2$ e $\partial z_2/\partial W_2$

1) $\dfrac{\partial \mathcal{L}}{\partial \hat{y}} = \dfrac{\hat{y} - y}{\hat{y}(1-\hat{y})}$ } Calcolata sopra

2) $\dfrac{\partial \hat{y}}{\partial z_2} = \dfrac{\partial \sigma(z_2)}{\partial z_2} = \sigma(z_2)\left(1-\sigma(z_2)\right) = \hat{y}\left(1-\hat{y}\right)$

3) $\dfrac{\partial z_2}{\partial W_2} = A_1^T$

$$\frac{\partial \mathcal{L}}{\partial W_2} = \frac{\hat{y} - y}{\cancel{\hat{y}(1-\hat{y})}} \cancel{\hat{y}(1-\hat{y})} A_1^T = (\hat{y} - y) A_1^T$$

2) Derivata su W_1

$$\frac{\partial \mathcal{L}}{\partial W_1} = \underbrace{\frac{\partial \mathcal{L}}{\partial z_2}}_{\hat{y} - y} \underbrace{\frac{\partial z_2}{\partial A_1}}_{W_2^T} \underbrace{\frac{\partial A_1}{\partial z_1}}_{\sigma'(z_1)} \underbrace{\frac{\partial z_1}{\partial W_1}}_{A_0^T}$$