

# Reti neurali

Vincenzo Bonifaci

IN550 – Machine Learning

# Una nuova occhiata alla regressione logistica

$$\Pr(y=1 | x) = \sigma(\underset{\uparrow}{w^T x})$$


Nella regressione logistica, abbiamo calcolato la probabilità che l'etichetta di  $x$  fosse  $y = 1$  come

$$\sigma(w^T x) = \sigma(w_0 + w_1 x_1 + \dots + w_d x_d)$$

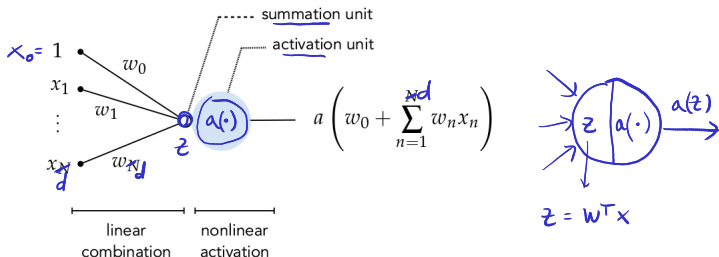
dove  $\sigma$  è la funzione **sigmoide**

# Neurone artificiale

Possiamo generalizzare questa operazione con un'unità *neurone artificiale*, che sulla base di stimoli  $(x_0, x_1, \dots, x_d)$  produce un valore di uscita

$$f = a \left( w^T x \right) = a \left( w_0 + w_1 x_1 + \dots + w_d x_d \right)$$

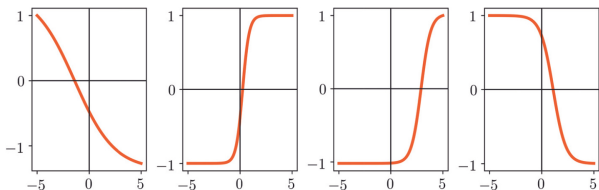
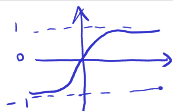
dove  $a : \mathbb{R} \rightarrow \mathbb{R}$  è un'opportuna *funzione di attivazione* (nonlineare) e il vettore  $w \in \mathbb{R}^{d+1}$  regola la forza delle *connessioni* dagli stimoli al neurone



## Altre funzioni di attivazione

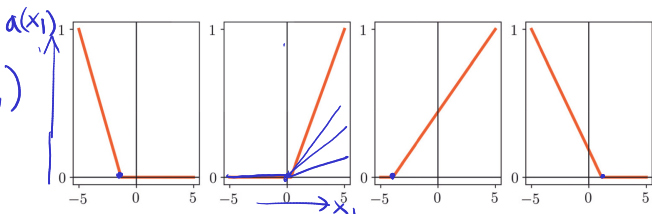
$$\frac{1 - 1}{1 + 1}$$

$$a(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$


 $w_0$   
 $w_1$ 

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 \\ x_1 \end{pmatrix}$$

$$w = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$



$$\Rightarrow w_0 = 0$$

$$\textcircled{w_1 = 1}$$

$$\Rightarrow z = x_1$$

$$a(z) = \max(0, x_1)$$

$$w^T x = w_0 + w_1 x_1 = z$$

$$a(z) = \max(0, z)$$

ReLU



# Esempio: emulazione di funzioni logiche

Sia  $a(z) = \max(0, z)$  (ReLU) e consideriamo due input  $x_1, x_2 \in \{0, 1\}$   
(True: 1, False: 0)

Allora il neurone

$$w = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}$$

$$z = w^T \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} \quad a(z) = ?$$

$$\rightarrow z = x_1 + x_2 - 1 \quad a(z) = \max(0, z)$$

equivale alla funzione AND( $x_1, x_2$ )

mentre il neurone

$$z = 1 - x_1 \quad a(z) = \max(0, z)$$

equivale alla funzione NOT( $x_1$ )

**Domanda 1.** Una unità ReLU può emulare la funzione OR( $x_1, x_2$ )?

**Domanda 2.** Se codifichiamo True con +1 e False con -1, un'unità con  $a(z) = \text{sgn}(z)$  può emulare le funzioni AND, OR e NOT?

# Strati di neuroni

Possiamo combinare  $U_1$  unità della forma

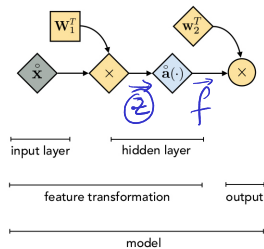
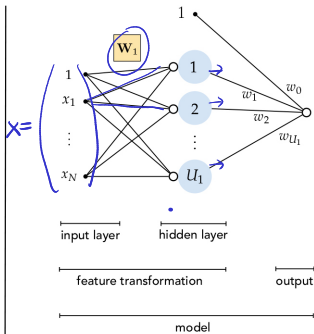
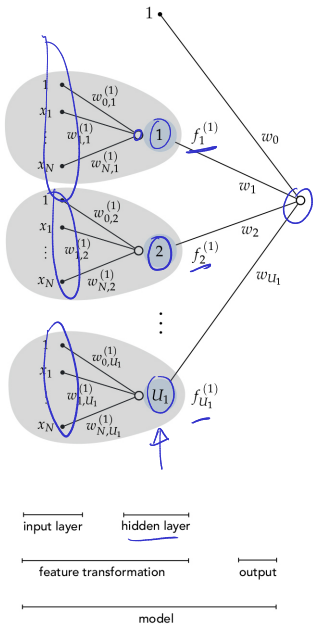
$$f_j^{[1]}(x) = a\left(w_j^{[1]\top} x\right) = a(z_j^{[1]})$$

per ottenere un output complessivo

$$w_0^{[2]} + w_1^{[2]} f_1^{[1]}(x) + \dots + w_{U_1}^{[2]} f_{U_1}^{[1]}(x) = W^{[2]\top} f^{[1]}$$

Tali  $U_1$  neuroni formano uno *strato nascosto* (hidden layer):

- non sono direttamente connessi uno all'altro
- lo strato è **nascosto** nel senso che il valore corretto che gli  $f_j^{[1]}(x)$  devono assumere per un dato esempio  $(x, y)$  non è noto (a differenza di quanto avviene per gli ingressi e l'uscita della rete)



# Esempio: ③ ingressi, $U_1 = 2$ , $U_2 = 1$

Chiamiamo lo strato di ingresso lo strato **zero**:

$$\underline{x_0 = 1}, \quad x_1 = f_1^{[0]}, \quad x_2 = f_2^{[0]}, \quad x_3 = f_3^{[0]}$$

Per le unità dello strato 1 (strato nascosto) abbiamo

$$z_1^{[1]} = W_1^{[1]T} x \quad f_1^{[1]} = a(z_1^{[1]})$$

$$z_2^{[1]} = W_2^{[1]T} x \quad f_2^{[1]} = a(z_2^{[1]})$$

dove  $W^{[1]}$  è una matrice  $2 \times (3 + 1)$  di parametri

Lo strato di uscita consiste di un solo neurone:

$$z_1^{[2]} = W_1^{[2]T} \underline{f^{[1]}} \quad f_1^{[2]} = a(z_1^{[2]})$$

dove  $W^{[2]}$  è una matrice  $1 \times (2 + 1)$  di parametri e  $f^{[1]}$  è il vettore calcolato dallo strato 1 (incluso il valore costante  $f_0^{[1]} = 1$ )

$$W^{[1]} = \begin{pmatrix} W_1^{[1]} \\ W_2^{[1]} \end{pmatrix} \in \mathbb{R}^4$$

$$W^{[2]} = \begin{pmatrix} W_1^{[2]} \end{pmatrix} \in \mathbb{R}^3$$



# Descrizione vettorizzata

In forma **vettorizzata** possiamo scrivere:

$$\underline{f^{[0]}} = (x_0, x_1, x_2, x_3)^T \in \mathbb{R}^4$$

$$\rightarrow \left[ \underline{z^{[1]}} = W^{[1]T} f^{[0]} \right] \quad \left[ f^{[1]} = \underline{a}(z^{[1]}) \right]$$

$$\underline{z^{[2]}} = W^{[2]T} \underline{f^{[1]}} \quad \underline{f^{[2]}} = \underline{a}(z^{[2]})$$

dove

$$\underline{a}(z_1, z_2, \dots) = (\underline{a}(z_1), \underline{a}(z_2), \dots)$$

La forma vettorizzata è cruciale per sfruttare appieno le risorse di calcolo disponibili: permette di sfruttare il parallelismo

# Reti neurali multistrato

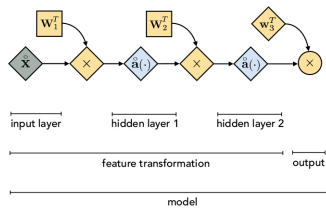
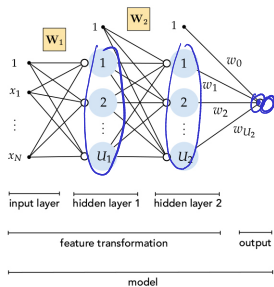
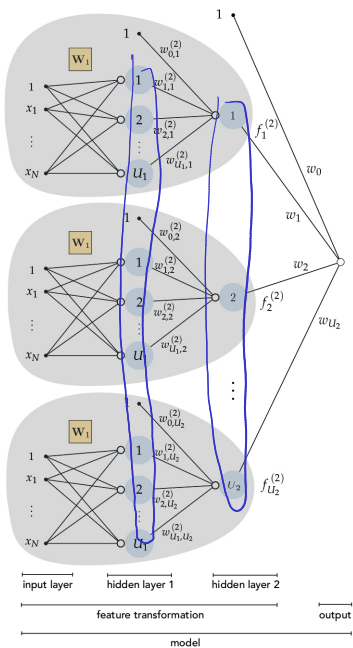
L'uscita di uno strato può fare da input per un secondo strato (nascosto) e così via:

$$h(x) = a \left( W^{[M]\top} a \left( W^{[M-1]\top} a \left( \dots a \left( W^{[1]\top} x \right) \right) \right) \right)$$

Aggiungendo strati:

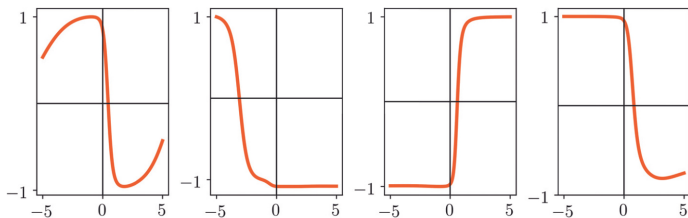
- La classe delle ipotesi rappresentabili si espande
- Aumenta la varianza (saranno necessari più esempi)

Il termine **deep** in *deep learning* si riferisce alla **profondità** delle reti neurali costruite e quindi alla potenziale complessità delle ipotesi apprese

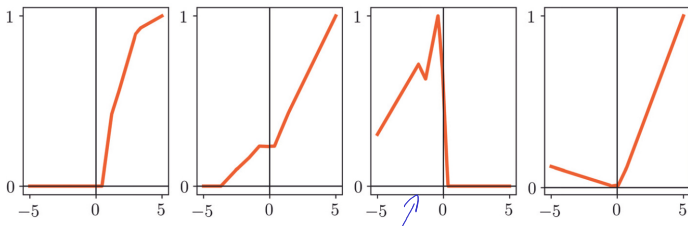


## Esempi di funzioni rappresentabili con reti a due strati

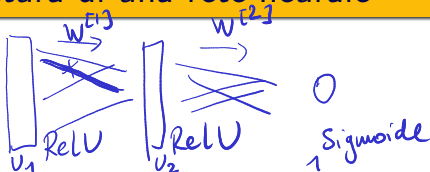
tanh



ReLU



# Parametri ed architettura di una rete neurale

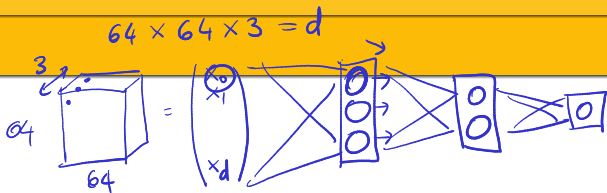


Una rete neurale è descritta da un'architettura e dei parametri:

- Architettura: numero di strati, numero di neuroni in ogni strato, tipo di funzioni di attivazione in ogni strato
- Parametri: una matrice  $W^{[k]}$  per ogni strato  $k$

Il tipo di rete qui discussa è detta *fully-connected* in quanto il neurone di ogni strato riceve un segnale da **tutti** i neuroni dello strato precedente

## Esempio



Una rete neurale deve classificare immagini RGB di dimensione  $64 \times 64$  ( $d = 64 \times 64 \times 3$ )

La rete ha la seguente architettura:

- 1 strato di input, 2 strati nascosti, 1 strato di output
- Rispettivamente  $(d, 3, 2, 1)$  unità in ciascuno strato
- Le funzioni di attivazione sono ReLU negli strati nascosti e sigmoide nello strato di output

$$64 \times 64 \times 3 + 1$$

$d$

$$3(d+1)$$

$$2(3+1)$$

$$1(2+1)$$

Quanti parametri ha la rete in tutto?

## Esempio (segue)

- Il primo strato nascosto ha  $3(d + 1)$  parametri (matrice  $W^{[1]}$ )
- Il secondo strato nascosto ha  $2(3 + 1)$  parametri (matrice  $W^{[2]}$ )
- Lo strato di output ha  $1(2 + 1)$  parametri (matrice  $W^{[3]}$ )
- In totale  $3d + 14 = 36878$  parametri

# Propagazione in avanti delle uscite

## Propagazione in avanti (*Forward propagation*)

L'uscita di ogni strato si ottiene *propagando in avanti* l'uscita dello strato precedente:

$$z^{[k]} = W^{[k]\top} f^{[k-1]} \quad f^{[k]} = a(z^{[k]})$$

fino ad ottenere l'uscita dell'ultimo strato



# Minimizzazione del rischio empirico nelle reti neurali

Le ipotesi delle reti neurali hanno la forma

$$\hat{y} = h(x) = f^{[M]}$$

dove  $M$  è l'indice dello strato di uscita della rete

Scegliendo una funzione di costo  $\ell$  arriviamo all'usuale rischio empirico

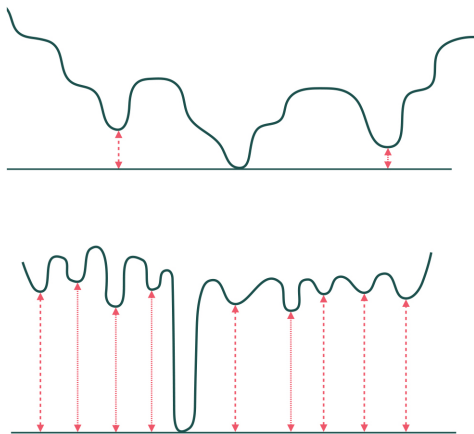
$$L_S(W) = \sum_{i=1}^m \ell(h, (x^{(i)}, y^{(i)}))$$

che cerchiamo di minimizzare scegliendo  $h$

Tipicamente:

- L'architettura della rete è fissata a priori
- Le matrici  $W^{[1]}, \dots, W^{[M]}$  sono oggetto dell'ottimizzazione
- Il problema di ottimizzazione risultante è **non convesso**

# Non convessità della funzione rischio empirico



In pratica, metodi del primo ordine avanzati (quali Gradiente normalizzato, RMSprop, Adam) forniscono spesso ottimi risultati

## Calcolo del gradiente: la *backpropagation*

Per implementare i metodi del primo ordine è sufficiente saper calcolare la derivata della funzione costo rispetto ad ogni matrice dei parametri:

$$\frac{\partial \ell}{\partial W^{[k]}}$$

Per far questo, possiamo sfruttare la definizione di  $f^{[M]}$  in termini di  $f^{[M-1]}$ , e applicare la regola della catena:

$$\frac{\partial \ell}{\partial W^{[k]}} = \frac{\partial \ell}{\partial f^{[M]}} \cdot \frac{\partial f^{[M]}}{\partial z^{[M]}} \cdot \frac{\partial z^{[M]}}{\partial f^{[M-1]}} \cdot \frac{\partial f^{[M-1]}}{\partial z^{[M-1]}} \cdot \dots \cdot \frac{\partial z^{[k]}}{\partial W^{[k]}}$$

# Calcolo del gradiente: la *backpropagation*

In generale,

Equazione di backpropagation

$$\begin{aligned} \frac{\partial \ell}{\partial W^{[k]}} &= \frac{\partial \ell}{\partial z^{[k+1]}} \cdot \frac{\partial z^{[k+1]}}{\partial f^{[k]}} \cdot \frac{\partial f^{[k]}}{\partial z^{[k]}} \cdot \frac{\partial z^{[k]}}{\partial W^{[k]}} \\ &= \frac{\partial \ell}{\partial W^{[k+1]}} \cdot \left( \frac{\partial z^{[k+1]}}{\partial W^{[k+1]}} \right)^{-1} \cdot \frac{\partial z^{[k+1]}}{\partial f^{[k]}} \cdot \frac{\partial f^{[k]}}{\partial z^{[k]}} \cdot \frac{\partial z^{[k]}}{\partial W^{[k]}} \end{aligned}$$

il che ci permette di calcolare  $\frac{\partial \ell}{\partial W^{[k]}}$  se conosciamo  $\frac{\partial \ell}{\partial W^{[k+1]}}$

⇒ Il gradiente viene calcolato dall'ultimo strato **a ritroso** (retropropagato) fino allo strato di input (*backpropagation*)