Funzioni di attivazione

Funzioni di attivazione

Premessa

Nell'ambito del ML e del DL, la funzione di attivazione *(fda)* è una funzione matematica utilizzata con il principale scopo di:

Introdurre non linearità agli output nella rete.

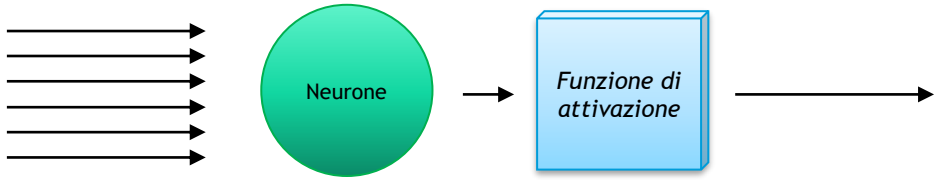
In generale:

- Esistono *fda* lineari e non lineari.
- Esistono molteplici *fda* per caso d'uso: ognuna con **pro** e **contro**.
- Possono essere applicate all'output dei neuroni.
- Possono essere applicate all'output dei layer.
- In genere si hanno *fda* per layer interni e per layer terminali.
- La *fda* dei layer interni è la stessa.

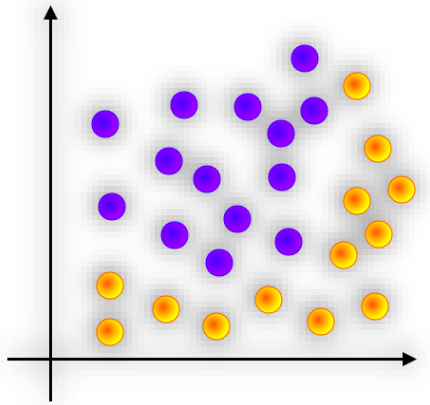
Funzioni di attivazione

La non-linearità

Quando si affronta un problema di ML/DL, si parla, in genere, di problemi difficili: problemi nei quali si cerca di estrarre *pattern complessi* dai dati.



Un semplice esempio, a spiegare il concetto, può partire dalla richiesta di separare due gruppi di dati nel piano cartesiano:

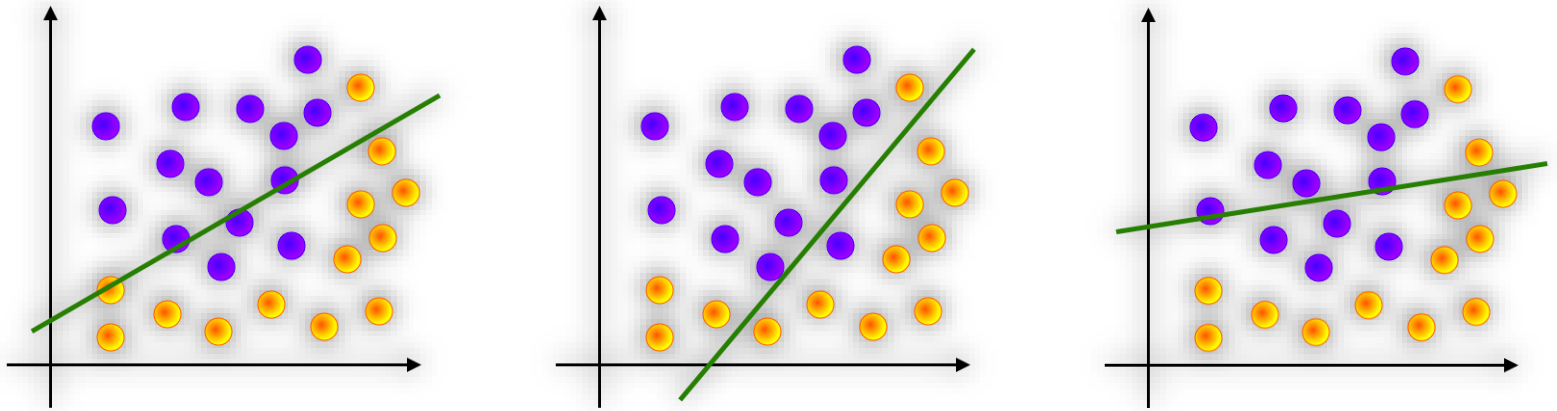


Funzioni di attivazione

La non-linearità

Il solo utilizzo della linearità porterebbe ad apprendere una «soluzione insoddisfacente» per la richiesta.

Questo è tanto più vero quanto più è complicato il modo in cui i dati vanno a interconnettersi fra loro…

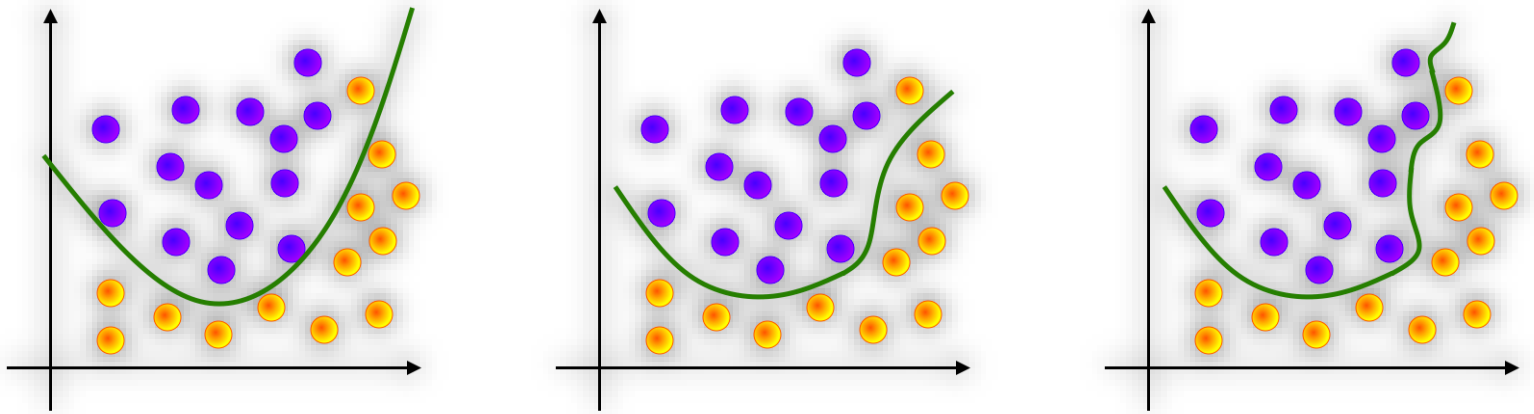


Funzioni di attivazione

La non-linearità

L'introduzione della non-linearità va vista come l'applicazione di un maggior potere di comprensione dei dati…

Della possibilità di «adattarsi» meglio ai dati e al problema stesso…



Di seguito, vedremo le principali e più note funzioni di attivazione.

Funzioni di attivazione

Linear

Range valori: $(-\infty, +\infty)$

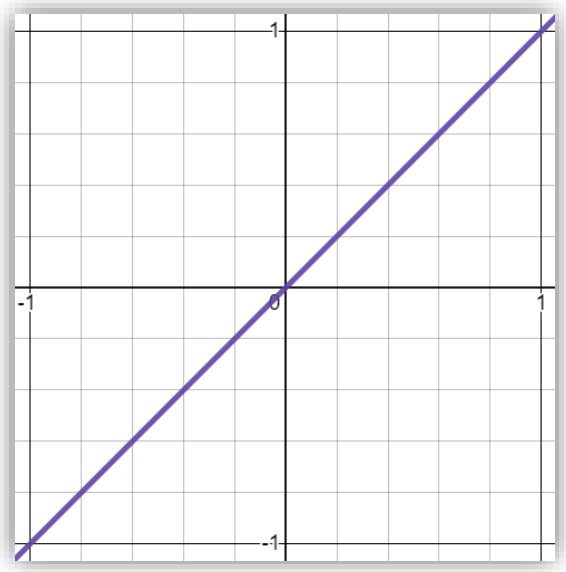
Pro:

- Semplice ed efficiente.
- Modifica l'input in maniera intuitiva.
- Adatta a problemi in cui è utile preservare la linearità e i range di valori non sono limitati.

Contro:

- La capacità espressiva è limitata al mondo lineare.
- Gradiente costante riduce le capacità di addestramento impedendo di riadattare la modifica dei pesi.

$y = x \qquad y = ax + b$



Funzioni di attivazione

Step

Range valori: $0 - 1$

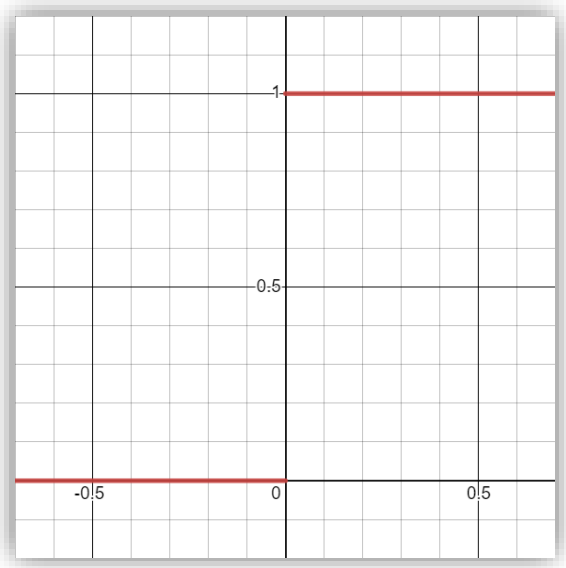
$$y = \begin{cases} 1 : x > 0 \\ 0 : x \leq 0 \end{cases}$$

Pro:

- Semplice ed efficiente.
- Adatta a problemi in cui è richiesto trovare una soglia di separazione interpretabile fra classi.

Contro:

- Non differenziabile nella soglia.
- Non granulare nell'output a fronte dell'input.
- Non permette di comprende quale input ha prodotto l'output e distinguere input più o meno «forti» a parità di output.



Funzioni di attivazione

Sigmoid, Logistic

$$y = \frac{1}{1 + e^{-x}}$$

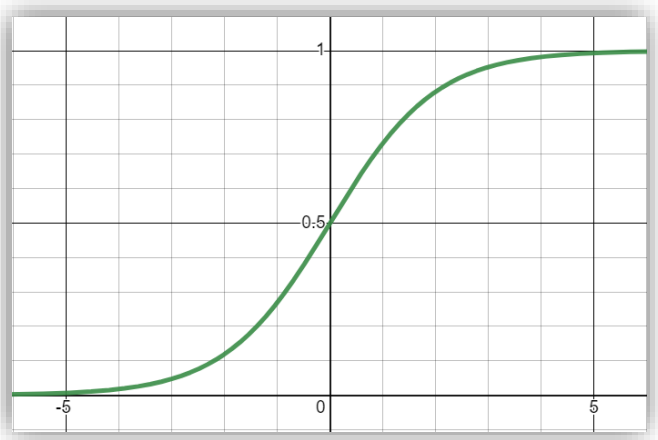
Range valori: $(0, 1)$

Pro:

- Permette di comprendere in maniera granulare come l'output è stato ottenuto a partire dall'input.
- Continua, differenziabile in ogni punto.
- Limitata nei valori di output.

Contro:

- La derivata, simmetrica in y, associa più valori allo stesso output.
- La derivata è informativa in un range ridotto.
- Propensa a problemi di «vanishing gradient».



Derivata: Wolfram

Utilizzata in generale nei layer finali ed in modelli che richiedono predizione di probabilità, ossia valori significativi fra 0 e 1.

Funzioni di attivazione

Tanh, Hiperbolic Tangent

$$y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

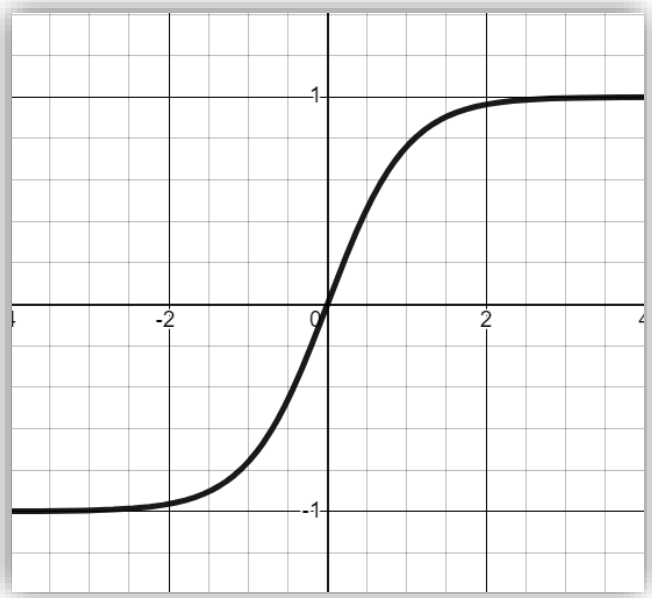
Range valori: $(-1, 1)$

Pro:

- Permette di comprendere in maniera granulare come l'output è stato ottenuto a partire dall'input.
- Continua, differenziabile in ogni punto.
- Limitata nei valori di output e centrata nello 0.

Contro:

- La derivata, simmetrica in y, associa più valori allo stesso output.
- La derivata è informativa in un range ridotto.
- Propensa a problemi di «vanishing gradient».



Derivata: Wolfram

Preferita nel caso di problemi in cui i dati, come la funzione, possono presentare **simmetrie** o essere **centrati nello zero**. Usata principalmente nei layer interni alla rete.

Funzioni di attivazione

ReLU

$$y = \max(0, x)$$

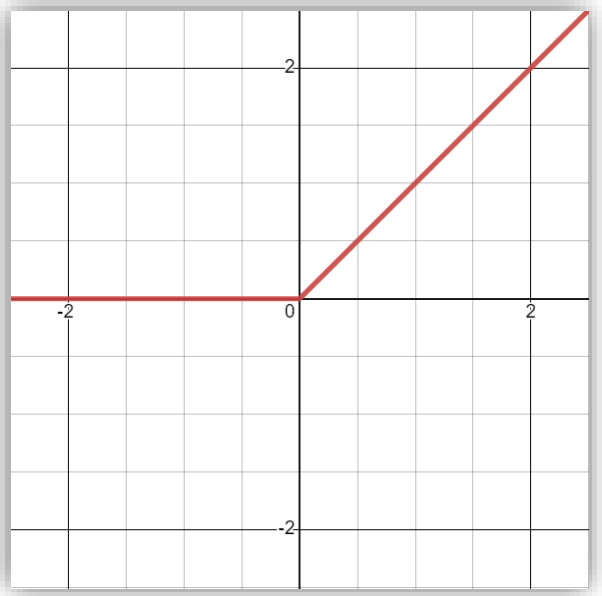
Range valori: $[0, +\infty)$

Pro:

- Di implementazione semplice ed efficiente.
- Mitiga il «vanishing gradient». (gradiente 1 per valori >0)
- Aumenta la sparsità dei dati grazie ad input negativi: efficienza computazionale, generalizzabilità e interpretabilità.

Contro:

- Gradiente 0 per valori ≤0: possibili neuroni non aggiornati.
- Non saturata per valori >0: possibile instabilità in addestramento.
- Annullando i valori ≤0, potrebbe presentare un bias nell'apprendimento da parte di valori positivi.



Utilizzata principalmente nei layer interni alla rete e grazie alle sue proprietà di semplicità ed efficienza.

Funzioni di attivazione

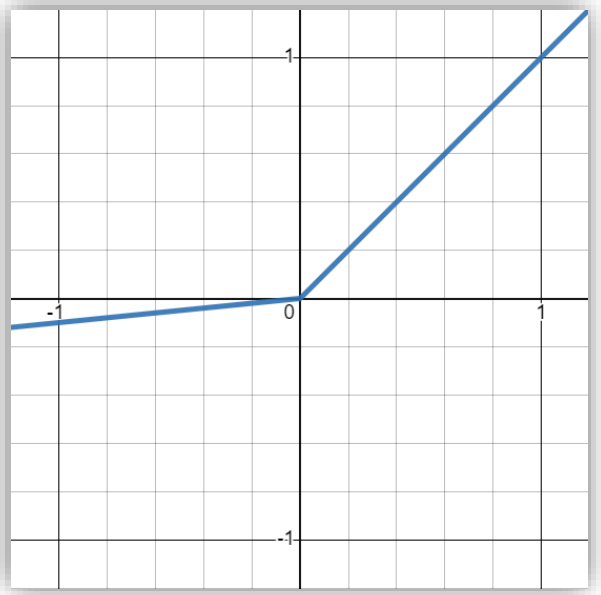
Leaky ReLU / Parametric ReLU

$$y = \begin{cases} ax & : x \leq 0 \\ x & : x > 0 \end{cases}$$

Range valori: $(-\infty, +\infty)$

Pro:

- Di implementazione semplice ed efficiente.
- Gradiente ≠ 0 per valori <0: superato il problema dei neuroni non aggiornati.
- Con una pendenza minima, non condensa l'informazione di gradiente dei valori <0 ad un solo valore (0).



Contro:

- Un possibile iper-parametro da scegliere, testare e valutare: la pendenza per la parte negativa.
- Per valori ridotti di *a* potrebbe presentare un bias nell'apprendimento da parte di valori positivi.

Cerca di risolvere i problemi della ReLU. Introduce un ulteriore iper-parametro da settare che: nel caso della *Parametric ReLU*, può venire appreso in addestramento.

Funzioni di attivazione

ELU

$$y = \begin{cases} a(e^x - 1) & : x \leq 0 \\ x & : x > 0 \end{cases}$$

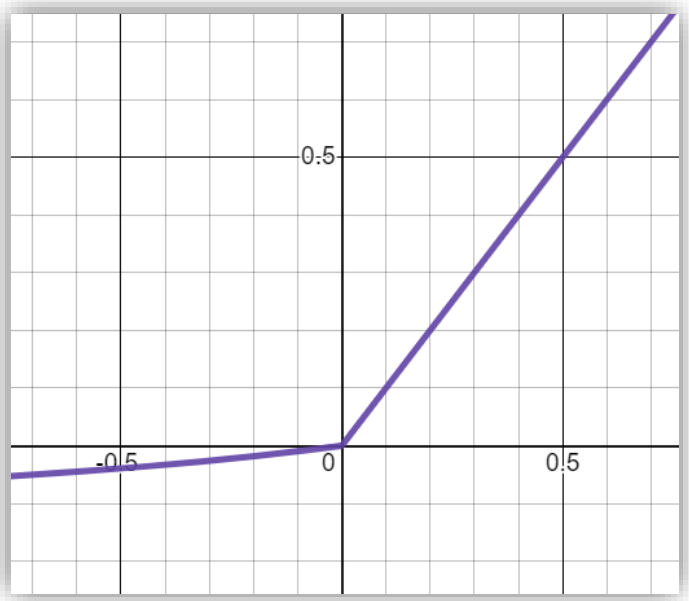
Range valori: $(-a, +\infty)$

Pro:

- Gradiente ≠ 0 per valori <0: superato il problema dei neuroni non aggiornati.
- Con la possibile saturazione dei valori negativi, riduce il bias legato alla predominanza di valori positivi.

Contro:

- Computazionalmente complessa, causa dell'esponenziale.
- Per valori sempre più negativi raggiunge il limite di saturazione, il gradiente tende a 0 e rallenta l'addestramento.



Nata per cercare di risolvere i problemi legati alla ReLU. Nonostante porti vantaggi, non è computazionalmente efficiente e va scelta con attenzione se le performance sono un vincolo importante.

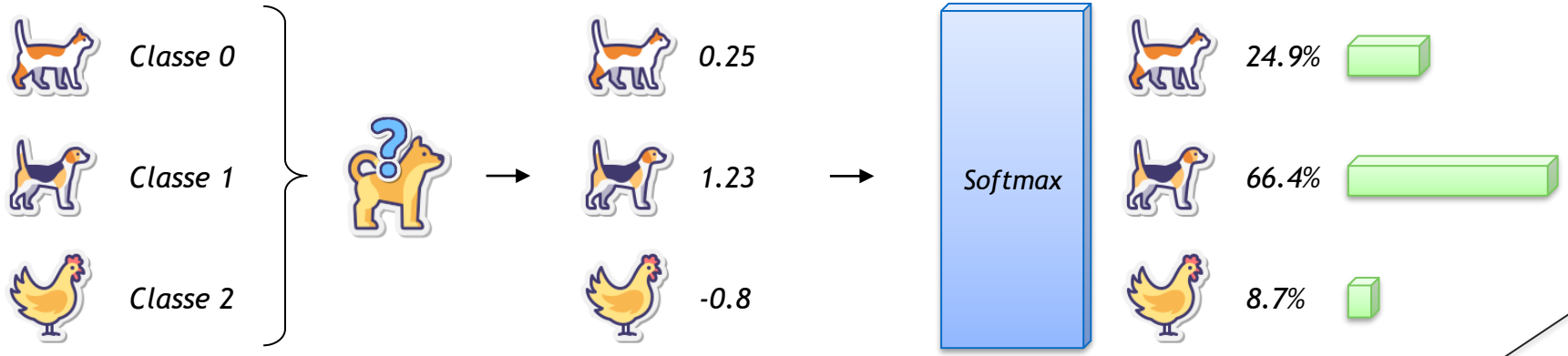
Funzioni di attivazione

Softmax

Questa funzione di attivazione ha lo scopo di convertire un vettore di valori numerici in un vettore di probabilità proporzionalmente ai valori stessi.

È utilizzata nei layer di output in reti per problemi di classificazione multi-classe.

$$y_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$



Rif: (Softmax)

Funzioni di attivazione

Riferimenti

Riferimenti PyTorch:

- *torch.nn — funzioni di attivazione 1*
- *torch.nn — funzioni di attivazione 2*

Riferimenti:

- *An Overview of Activation Functions*
- *Activation Functions in Neural Networks*

Proviamo?