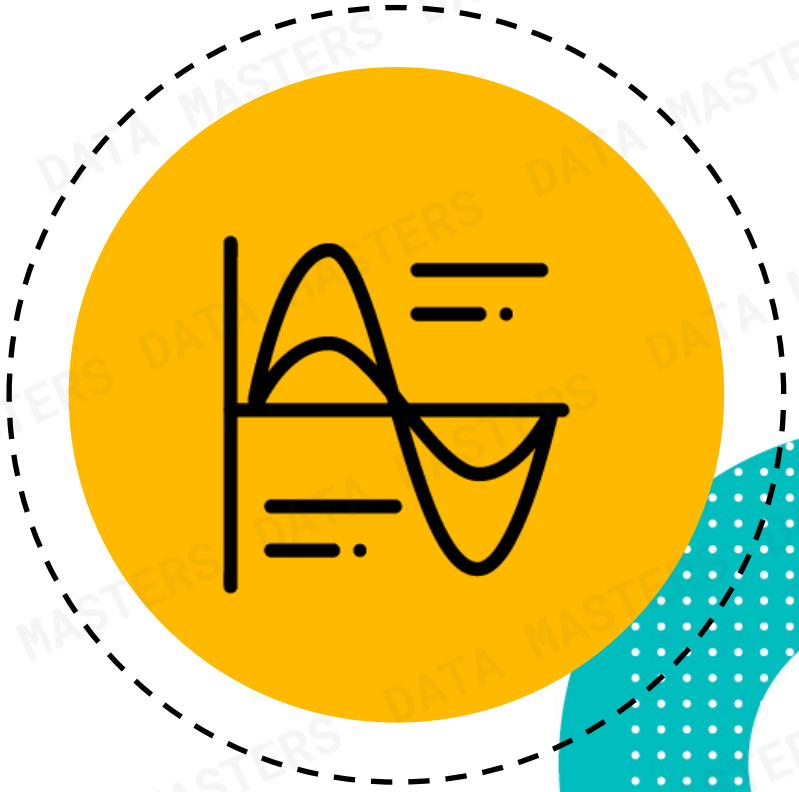
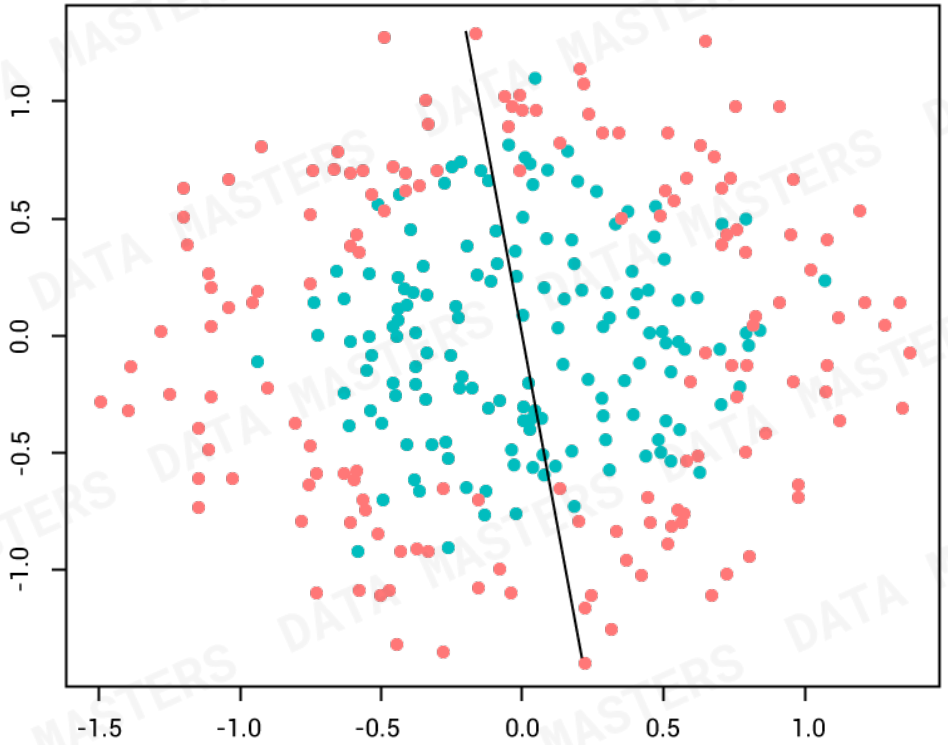
Funzione di attivazione

- Note anche come **funzione di trasferimento**
- **mappano** i nodi di input sui nodi di output
- utilizzate per inserire **non linearità**
- simile a quello che accade nei **neuroni biologici**:
  il potenziale d'azione viene trasmesso integralmente una volta che la differenza di potenziale alle membrane supera una certa soglia

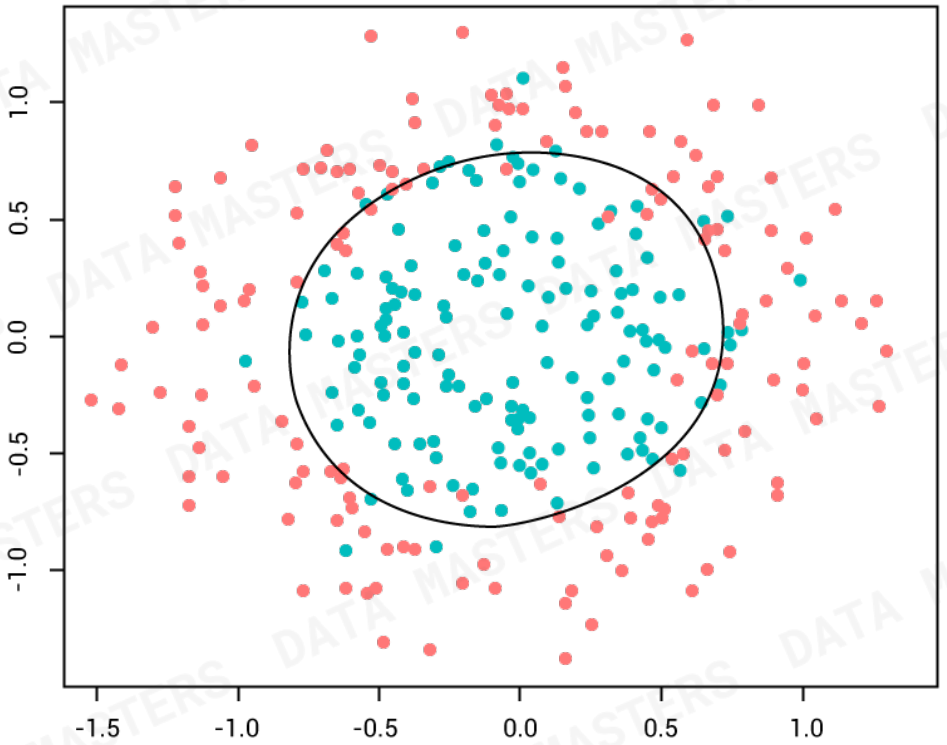


Funzione di attivazione

Funzione di Attivazione Lineare

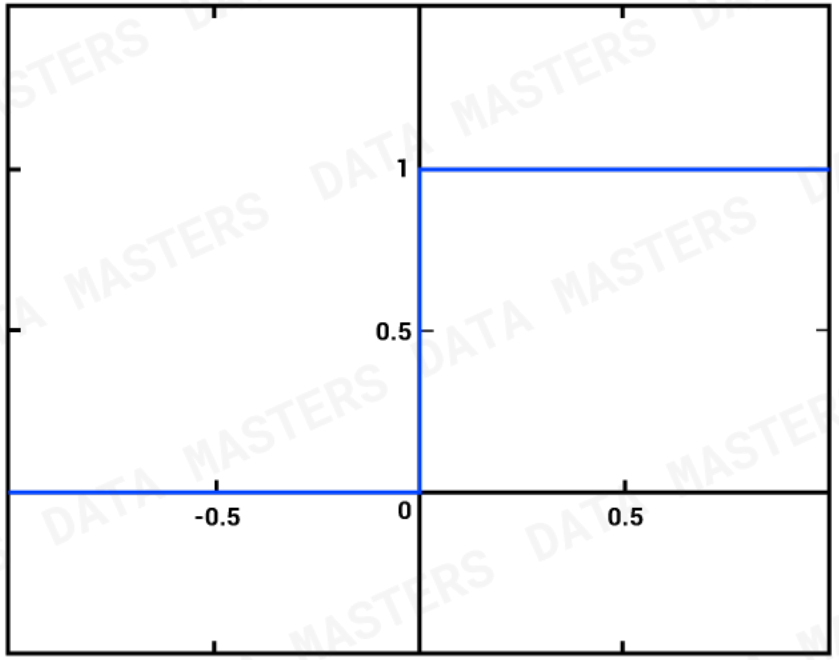


Funzione di Attivazione non Lineare



Step Function - Funzione di attivazione a gradino

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



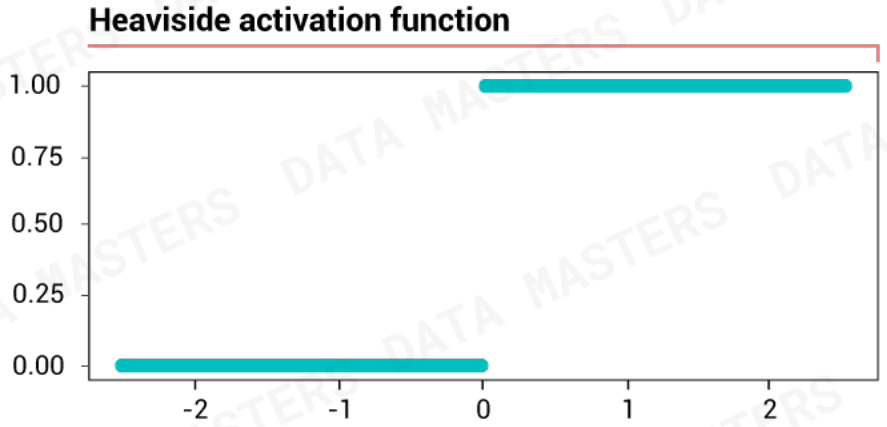
- Facile da calcolare
- Normalizza l'output tra 0 e 1
- Ha un output binario
- Non utilizzata con la discesa del gradiente perché ha derivata = 0 dovunque eccetto che in zero dove ha una discontinuità e non è derivabile

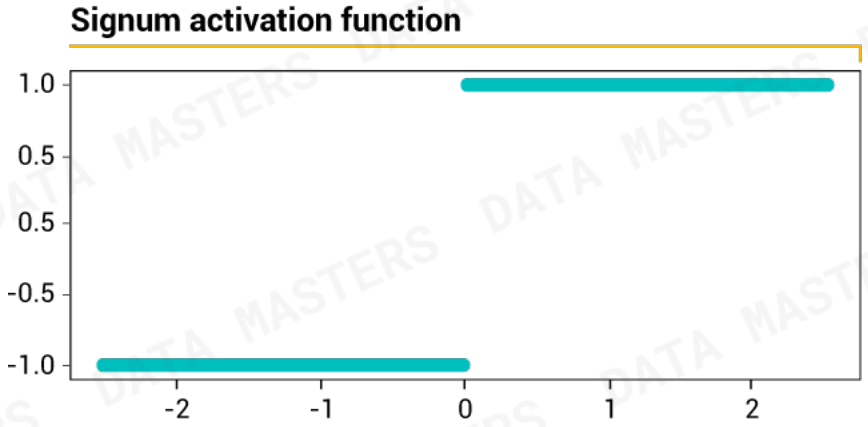
Step Function – varianti

Utilizzate in varianti del **Percettrone**

$$\begin{cases} y=1 & \text{per } x>0 \\ y=0.5 & \text{per } x=0 \\ y=0 & \text{per } x<0 \end{cases}$$

Heaviside

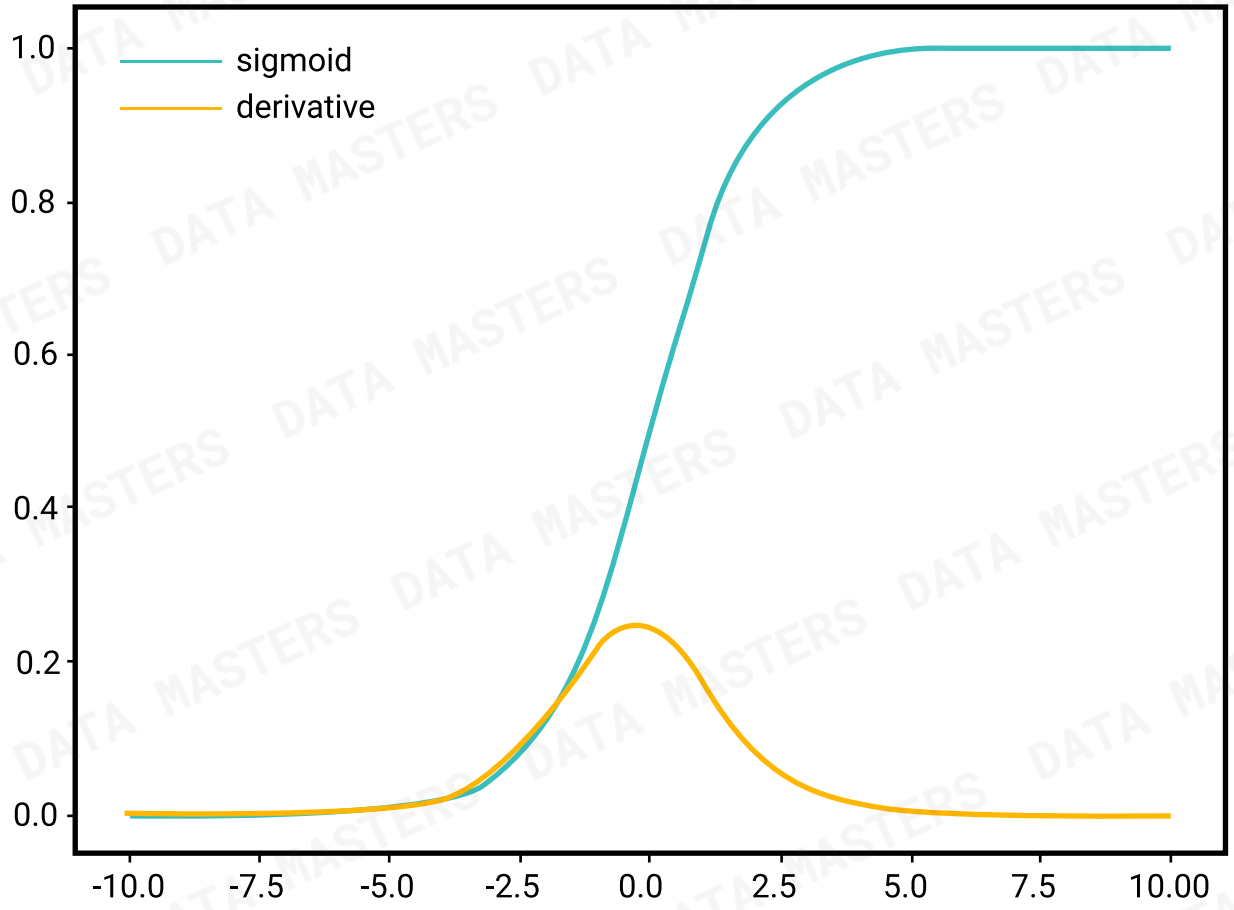




Signum

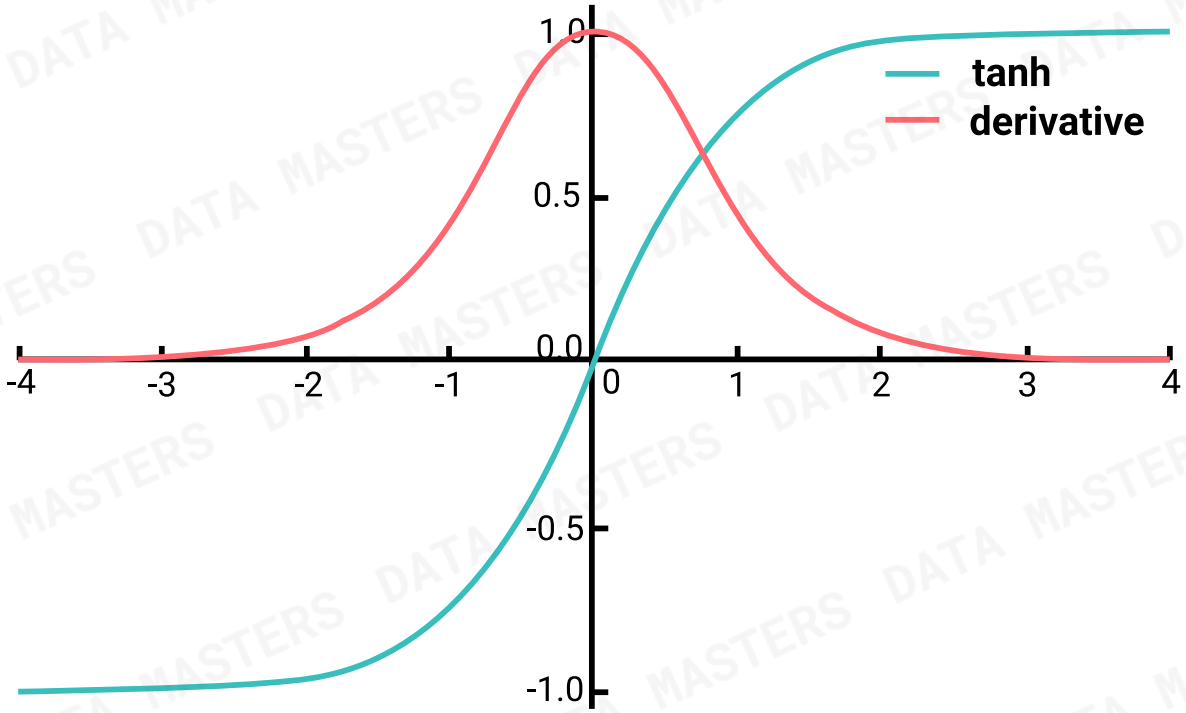
$$\begin{cases} y=1 & \text{per } x>0 \\ y=0 & \text{per } x=0 \\ y=-1 & \text{per } x<0 \end{cases}$$

Funzione di attivazione Sigmoide



- Passaggio graduale per y tra 0 e 1
- Ad x grandi e piccole ha derivata prossima a zero, **«svanisce»** il gradiente, convergenza lenta
- Non ha la y centrata sullo zero: valori di ogni step di learning solo positivi o negativi
- Aggiunge non linearità ad un modello

Funzione di attivazione Tangente Iperbolica

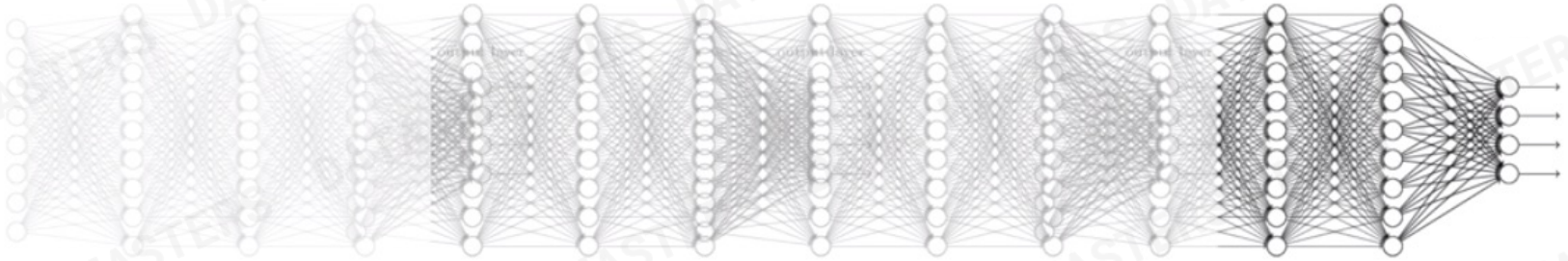


- Passaggio graduale per y tra -1 e 1
- Ad x grandi ha derivata prossima a zero, **«svanisce»** il gradiente, convergenza lenta
- Ha la y centrata sullo zero: valori di ogni step di learning positivi e negativi, apprendimento più veloce rispetto alla sigmoide
- Aggiunge non linearità ad un modello

$$tanh(x) = \frac{2}{1+e^{-2x}} - 1 = 2\,sigmoid(2x) - 1$$

Scomparsa del gradiente

E’ una difficoltà riscontrata nell’apprendimento automatico tramite discesa del gradiente



Ogni peso della rete viene aggiornato sfruttando la derivata prima parziale della funzione di costo rispetto al peso stesso. Se il gradiente risulta in un valore prossimo allo zero, il relativo peso non viene aggiornato

Problema opposto: esplosione del gradiente

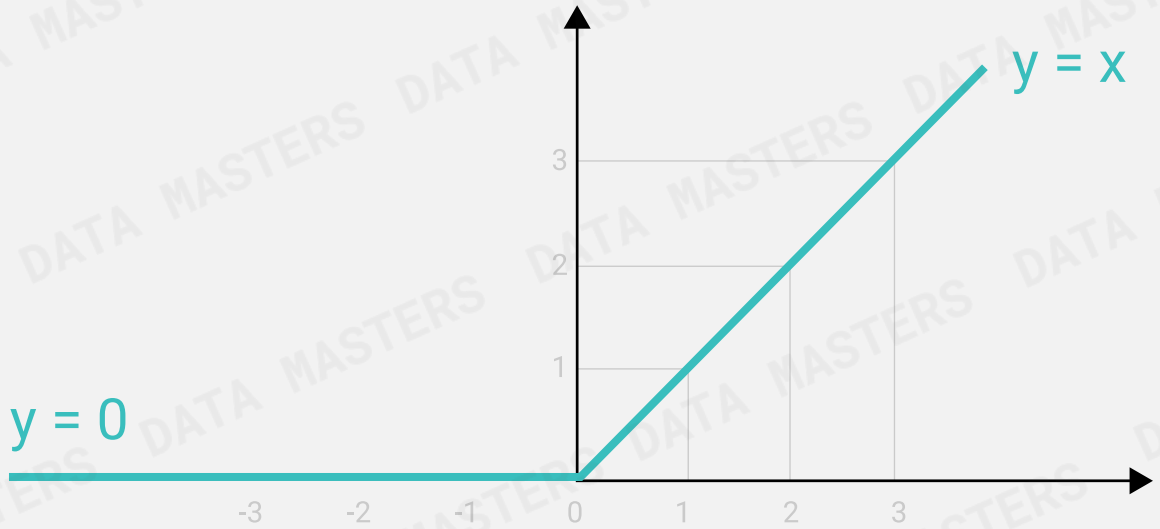
Funzione di attivazione ReLu (Rectifier Linear Unit)

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

derivata = 0 per $x<0$

derivata = 1 per $x>0$

derivata = 0 per $x=0$ (per definizione)



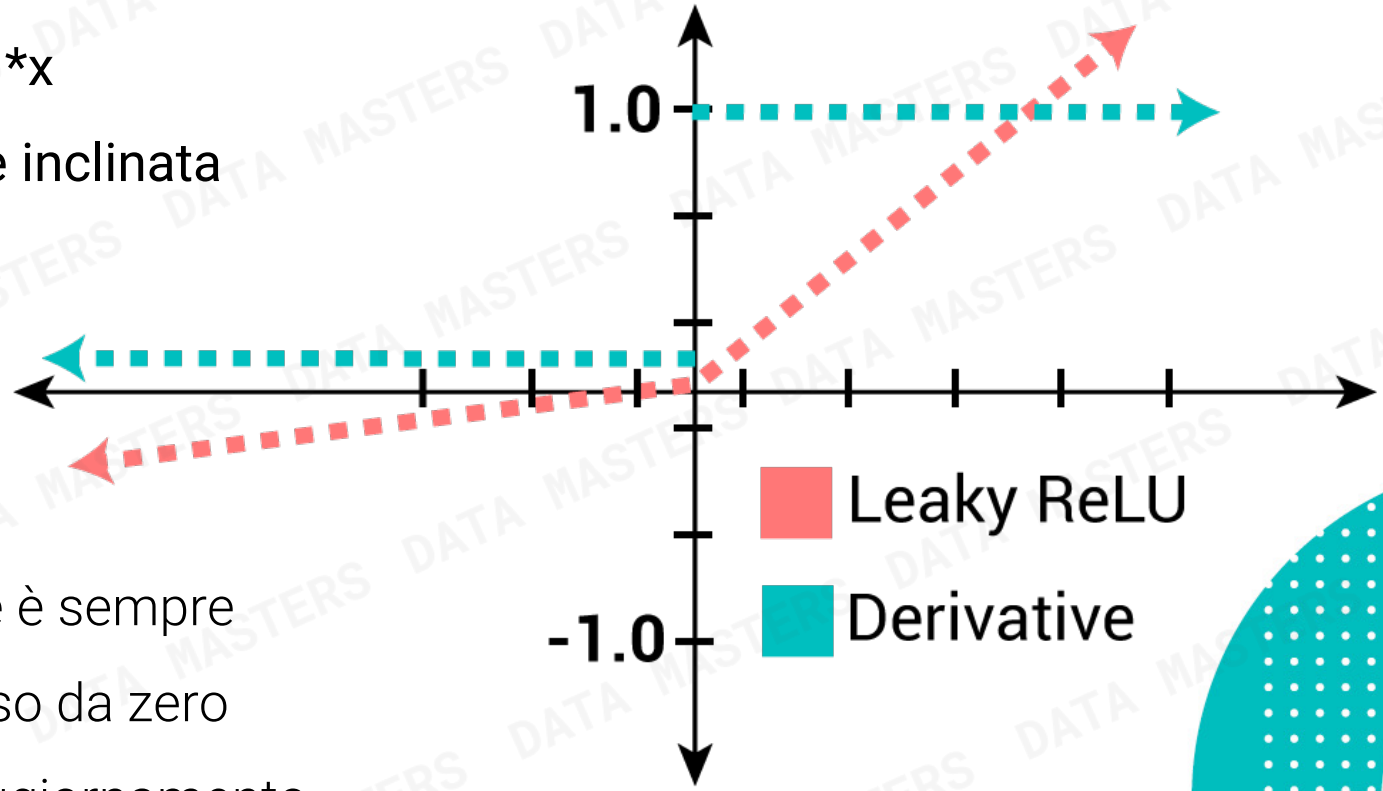
- Riduce la scomparsa del gradiente
- Dead neurons
- Largamente utilizzata, specialmente nei layer intermedi
- Più veloce da calcolare rispetto a sigmoide e tangente iperbolica

Funzione di attivazione Leaky ReLu

La funzione Leaky ReLU imposta y=0,01*x

per x < 0 creando una linea leggermente inclinata

(con derivata piccola ma ≠ 0)



Esistono diverse varianti la cui idea base è sempre

fare in modo di avere un gradiente diverso da zero

ed eventualmente recuperare il basso aggiornamento

dei pesi utilizzando più epoche di addestramento