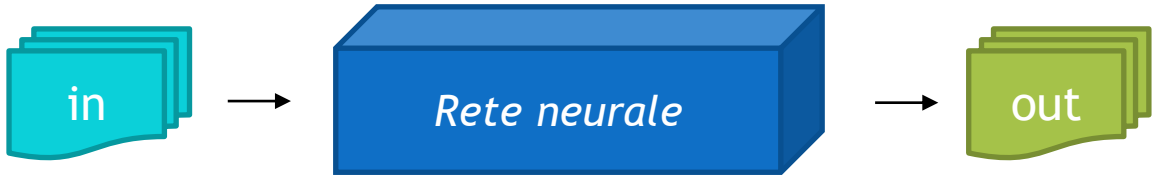
Autograd

Autograd

Premessa

L'obbiettivo di una rete neurale è, in termini assoluti, uno solo:

Imparare al meglio cosa porta un ***input*** ad essere l'***output*** desiderato.



Apprendere significa scoprire una relazione…imparare quali regole governano un particolare contesto, dal più complicato al più semplice.

Quante uova servono per una semplice omelette? 2!

Autograd

Premessa

Il sistema utilizzato per apprendere è la creazione di una rete interconnessa fatta di centinaia, migliaia…miliardi di nodi in comunicazione fra loro.

Possiamo trovare:

- Nodi di input.
- Nodi che manipolano, nodi operazione.
- Nodi di output.

Ognuno con le proprie caratteristiche e «manopole» (i suoi parametri) a regolarne il comportamento.

Autograd

Premessa

Lo stesso si trova in **PyTorch** dove un nodo può rappresentare:

- Un'operazione o una combinazione di operazioni.
- Un parametro da regolare.
- Un semplice operando.

Per questo, nel definire una rete, questo è ciò che accade:

- Si definiscono gli input e gli output e le operazioni che li manipolano.
- Ne nasce un grafo che traccia l'intero percorso.

Ne risulta, quindi, un grande riassunto:

il grafo computazionale

Autograd

Proprietà nei nodi

Alla creazione di un tensore in PyTorch, gli si vanno ad assegnare alcune caratteristiche importanti, in particolare:

- Il ruolo che il tensore avrà nel grafo.
- Il modo in cui è stato creato.

Più nello specifico, saranno esposte proprietà quali:

is_leaf	Indica nodi per i quali saranno calcolati i gradienti.
requires_grad	Indica se è richiesto il calcolo di gradienti per il nodo.
grad	Accumula un tensore di gradienti per il nodo di appartenenza
grad_fn	Memorizza quale operazione ha generato il nodo corrente.

Autograd

Proprietà nei nodi

Alla creazione di un tensore da parte di un utente, può essere quindi richiesto di calcolarne, durante la back-propagation, il gradiente o meno:

Tensori utente con ***requires_grad*** a **False** sono foglie.

Tensori utente con ***requires_grad*** a **True** sono foglie.

Tensori foglia per cui è richiesto il calcolo del gradiente, lo accumuleranno durante la fase di back-propagation.

Autograd

Proprietà nei nodi

Ai tensori vi si applicano operazioni al fine di manipolarli, modificarne il comportamento e realizzare sistemi complessi…reti neurali ad esempio.

Tensori nati da operazioni con ***requires_grad*** a **False** rimangono foglie.

*Per questi tensori non foglia, il gradiente non è accumulato di default ma va richiesto appositamente. (**retain_grad**)*

Tensori nati da operazioni con ***requires_grad*** a **True** non sono foglie.

Al momento della creazione tracciano però l'operazione che li ha creati, gli operanti entrati in gioco e come rintracciarli muovendosi nel grafo.

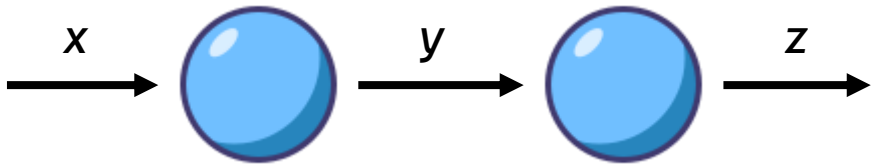
Autograd

Chain-Rule

Il sistema automatico di calcolo del gradiente, sfrutta una proprietà fondamentale chiamata **chain-rule**:

$$\frac{d(f(g(x))}{dx} = f'\big(g(x)\big) * g'(x)$$

Il un sistema, il grafo, fatto di nodi ed operazioni, questo concetto rimane valido e ne vede la massima espressione:



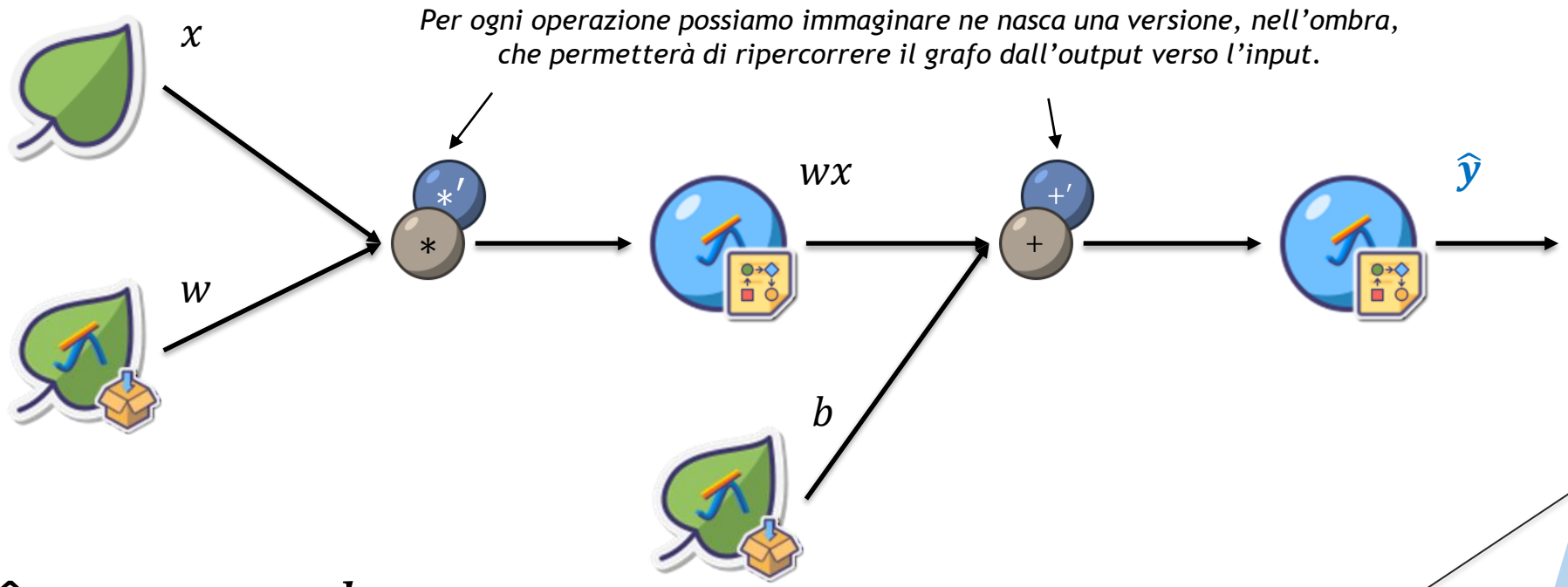
$$\frac{dz}{dx} = \frac{dz}{dy} * \frac{dy}{dx}$$

*L'influenza di **x** in **z** è l'influenza che **x** ha posto in **y** ed **y** in **z**.*

Autograd

Piccolo esempio

Volendo apprendere una trasformazione dell'input, il grafo computazionale avrebbe una forma simile a questa:

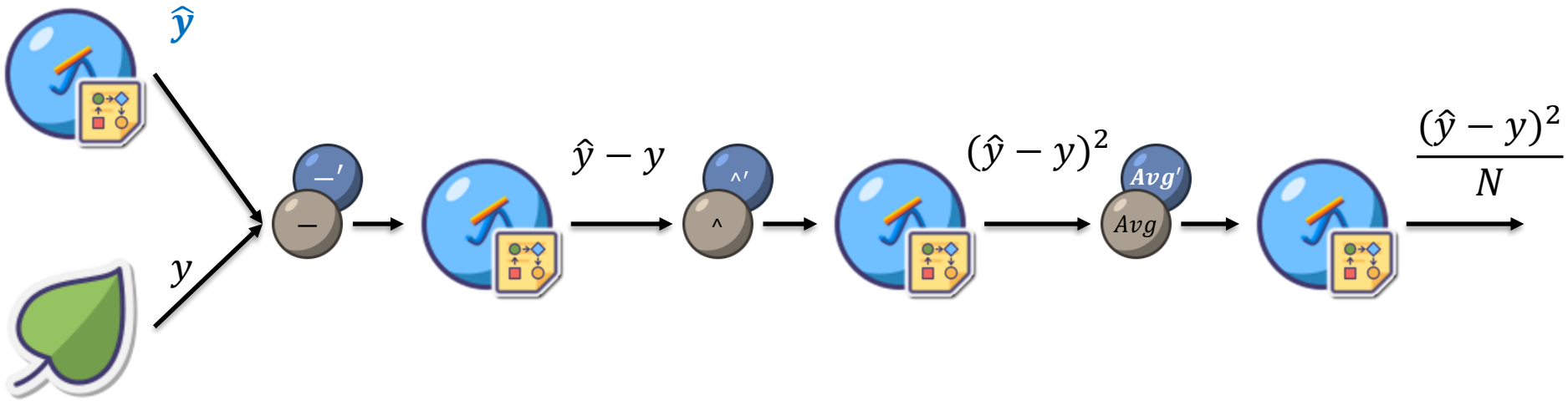


$$\hat{y} = w * x + b$$

Autograd

Piccolo esempio

Volendo calcolare l'errore commesso, la *loss*, come '*mean squared error*' rispetto al valore atteso, avremmo:

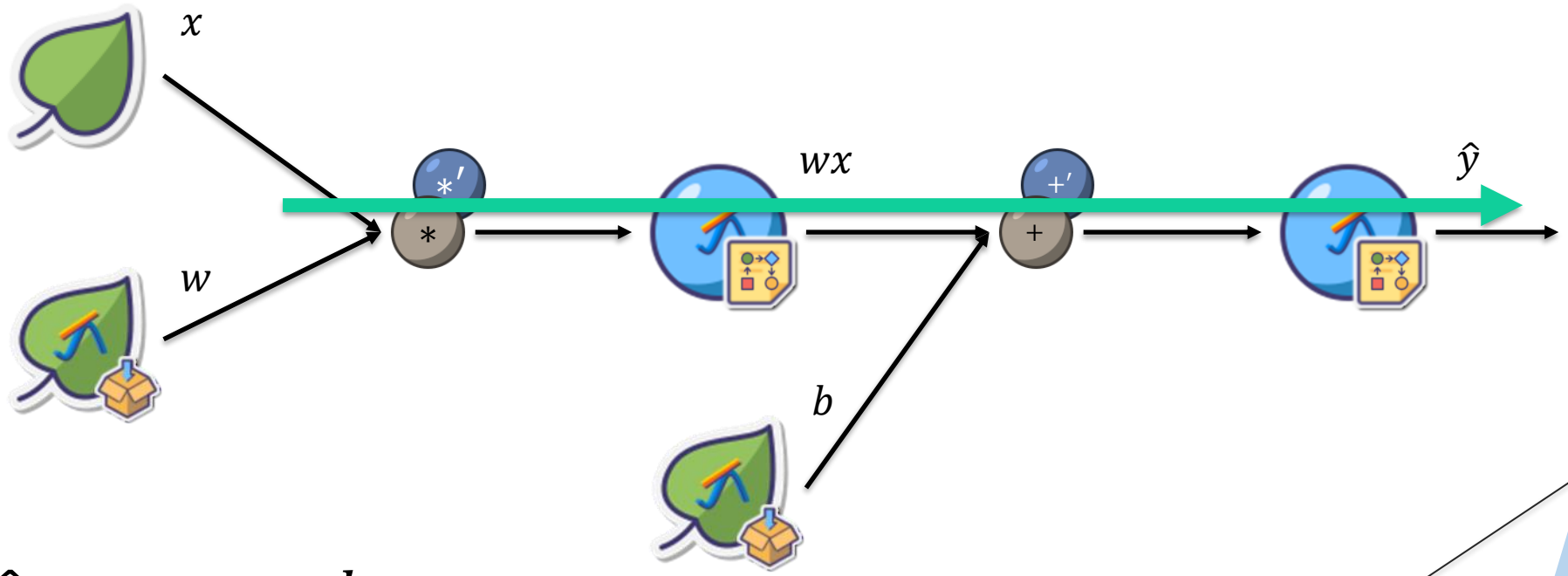


$$loss = \frac{(\hat{y} - y)^2}{N}$$

Autograd

Piccolo esempio: forward pass

Nel forward pass gli input viaggiano e si combinano con i parametri addestrabili: inizializzati al primo *step* e successivamente, di volta in volta, modificati.

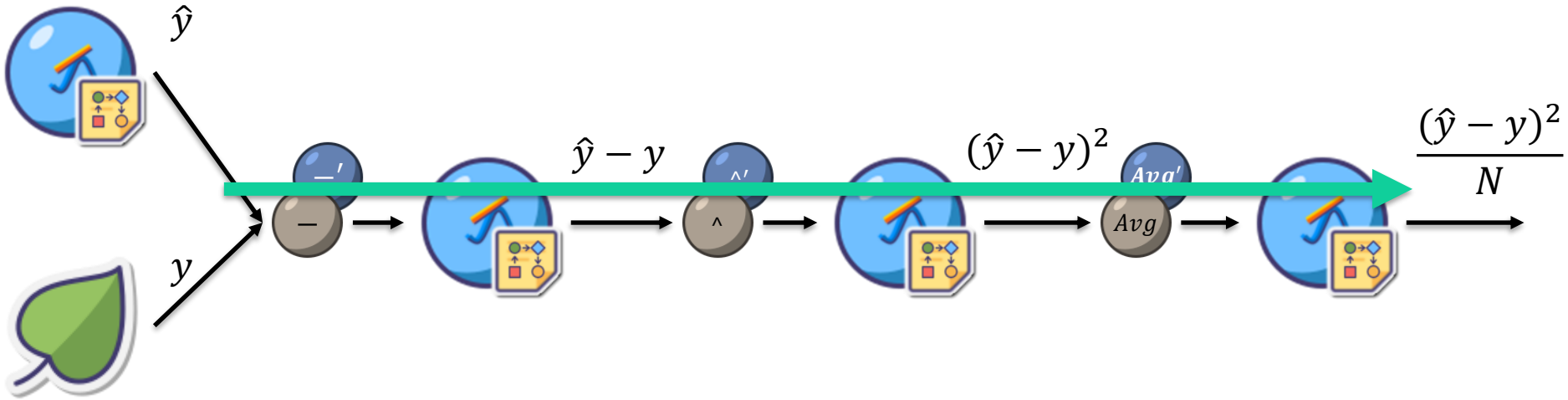


$$\hat{y} = w * x + b$$

Autograd

Piccolo esempio: forward pass

Nel forward pass gli input viaggiano e si combinano con i parametri addestrabili: inizializzati al primo *step* e successivamente, di volta in volta, modificati.

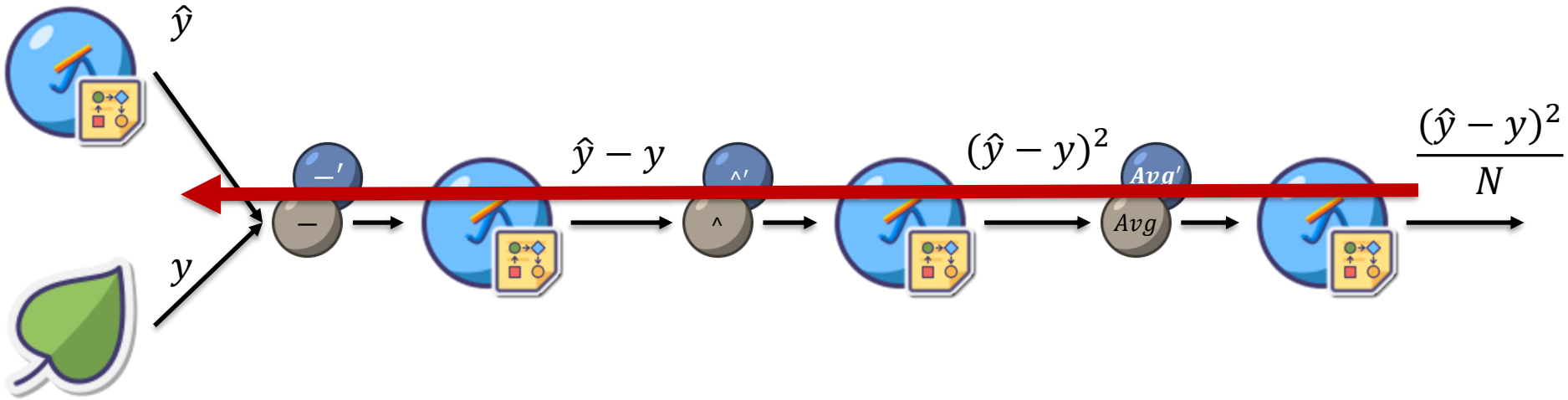


$$loss = \frac{(\hat{y} - y)^2}{N}$$

Autograd

Piccolo esempio: back propagation

Durante la backpropagation si cerca come i parametri addestrabili hanno influenzato l'output finale.

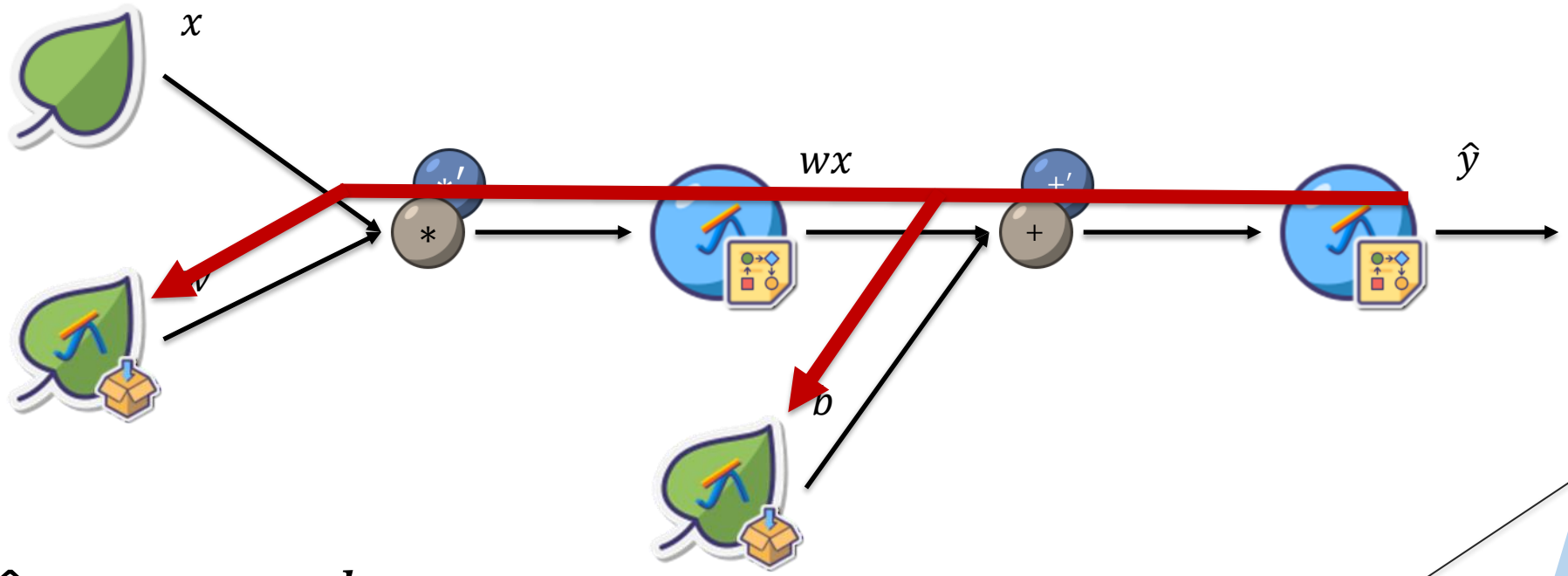


$$loss = \frac{(\hat{y} - y)^2}{N}$$

Autograd

Piccolo esempio: back propagation

Durante la backpropagation si cerca come i parametri addestrabili hanno influenzato l'output finale.



$$\hat{y} = w * x + b$$

Proviamo?