Ottimizzatori

Ottimizzatori

Premessa

L'ottimizzatore è l'elemento chiave del processo di apprendimento.

È usato per **aggiornare i parametri** (i *pesi*) della rete in modo da minimizzare o massimizzare la *loss* (in genere definita come la differenza tra predizione e valore atteso).

Si occupa di:

- Calcolare i gradienti della funzione di costo rispetto ai parametri della rete durante la fase di back-propagation.
- Calcolare di quanto i pesi debbano essere aggiornati.
- Aggiornare i pesi della rete.

Ottimizzatori

Premessa

La scelta del corretto ottimizzatore influisce sull'esito dell'addestramento in quanto:

*Scegliere l'ottimizzatore implica scegliere l'algoritmo che si occuperà di calcolare di **quanto** i pesi debbano essere aggiornati, e **come** verranno aggiornati.*

Ogni algoritmo ha le sue peculiarità, vantaggi e svantaggi:

- *Stochastic Gradient Descent (SGD)*
- *SGD with momentum*
- *Adaptive Gradient (AdaGrad)*
- *RMSProp*
- *Adam*
- *AdaDelta*

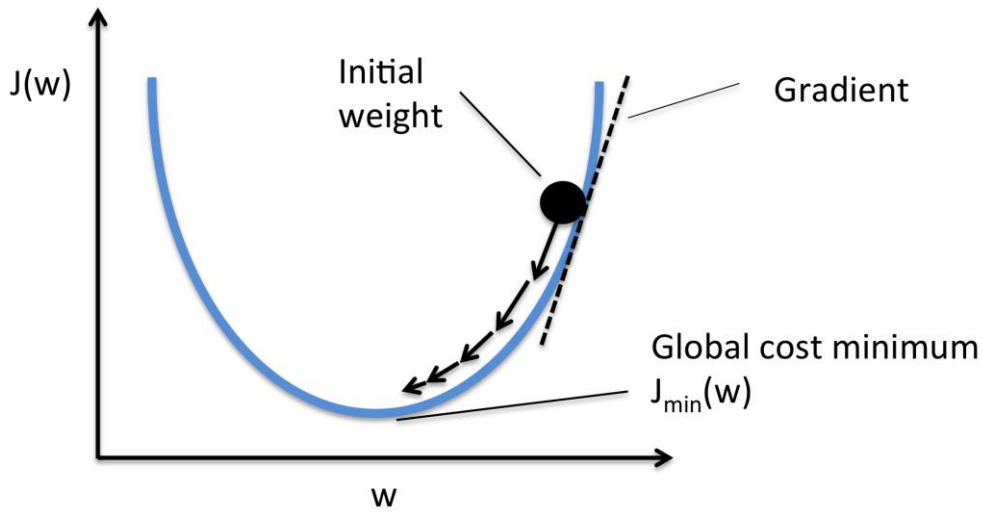
Ottimizzatori

Gradient Descent

Algoritmo base, con cui vengono aggiornati i parametri della rete.

Calcola il gradiente della loss rispetto ai pesi del modello sfruttando l'intero set di dati.

Aggiorna i pesi muovendosi nella direzione opposta a quella del gradiente.



Peso aggiornato : W_{new}
Peso originale : W_{old}
Learning rate : α

$$W_{new} = W_{old} - \alpha * \frac{\partial(Loss)}{\partial(W_{old})}$$

Ottimizzatori

Stochastic Gradient Descent (SGD)

L'algoritmo Gradient Descent esegue calcoli tanto più onerosi quanto più la raccolta dati è grande.

Stochastic Gradient Descent, basato su GD, risolve questo problema:

*Calcola il gradiente della loss rispetto ai parametri del modello su **un campione alla volta**.*

In questo modo i parametri vengono aggiornati molto di frequente: *per ogni singolo campione*.

Ottimizzatori

Mini-Batch Stochastic Gradient Descent

Basato su SGD, l'algoritmo a *mini-batch* calcola il gradiente della loss rispetto ai parametri del modello usando **una raccolta di campioni** selezionata randomicamente.

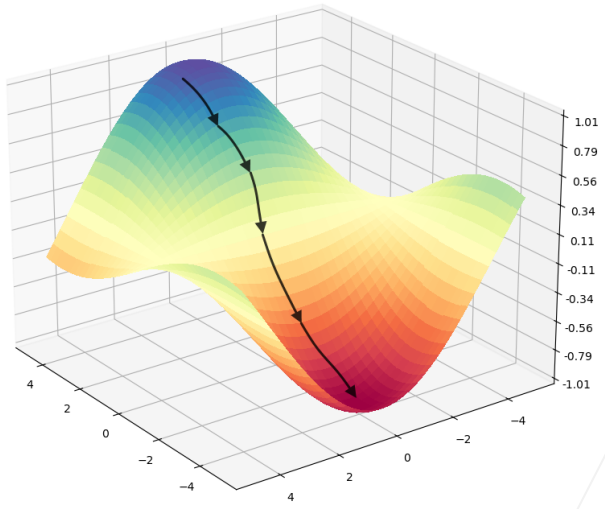
Gli aggiornamenti vengono poi eseguiti per ogni batch di dati.

Mini-batch SGD migliora GD ed SGD nei seguenti aspetti:

- Riduce il numero di calcoli rispetto SGD.
- Supera i problemi del GD.
- Combina i benefici di SGD e GD.

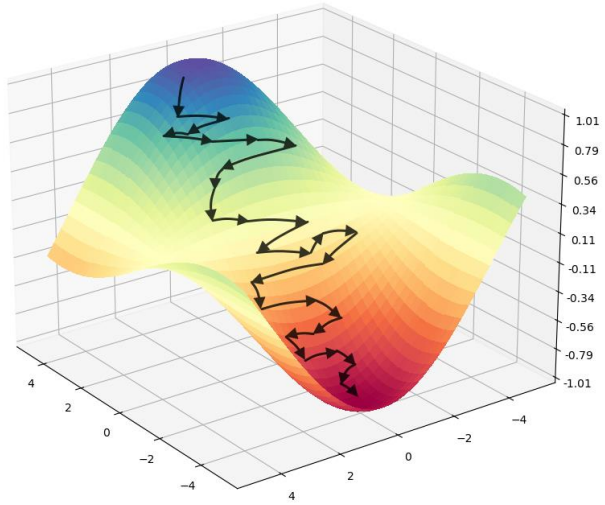
Ottimizzatori

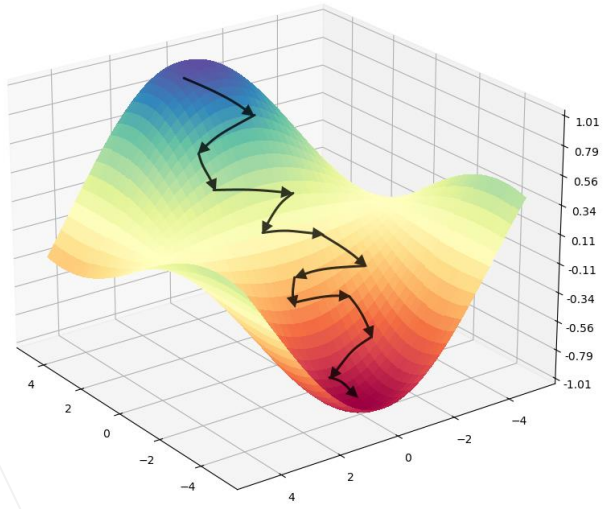
GD, SGD e Mini-Batch SGD (confronto grafico)



Gradient Descent

Stochastic Gradient Descent





Mini-Batch SGD

Ottimizzatori

GD, SGD e Mini-Batch SGD: pro e contro

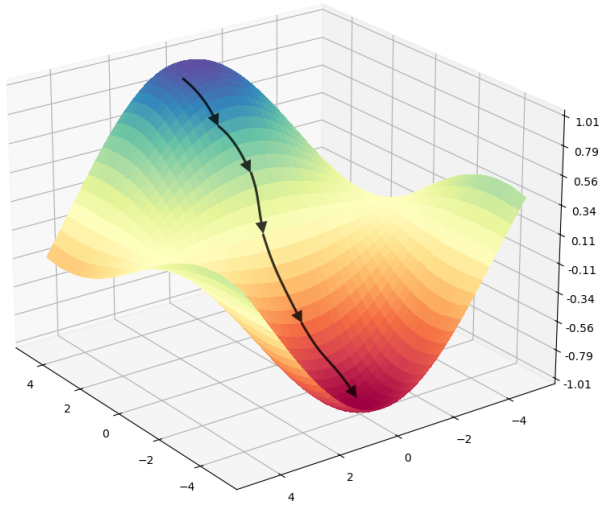
Gradient Descent, GD:

Pro:

- Efficiente nell'ottimizzare e parallelizzabile.
- Adattabile a una vasta gamma di funzioni di costo.

Contro:

- Sensibile alla partenza: rischio di convergere a minimi locali o punti di sella.
- Richiede di mettere a punto iperparametri come il tasso di apprendimento.
- Possibilità di overfitting, specialmente con dataset complessi.
- Il peso in memoria cresce con il dataset.



Gradient Descent

Ottimizzatori

GD, SGD e Mini-Batch SGD: pro e contro

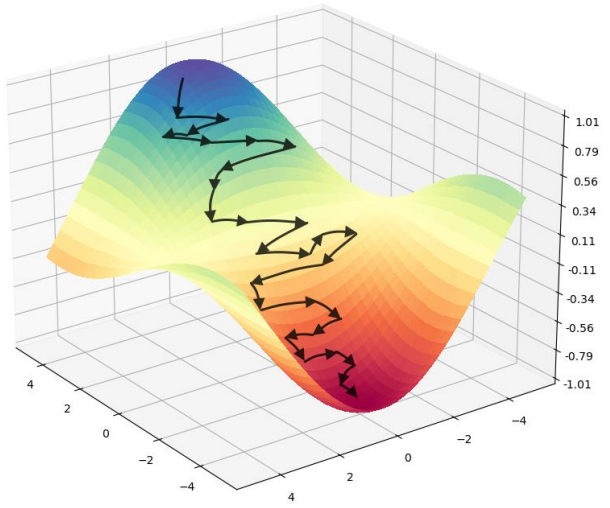
Stochastic Gradient Descent, SGD:

Pro:

- Veloce ad aggiornare i pesi del modello.
- Reattivo ai cambiamenti nei campioni.
- Ridotta richiesta di memoria, adatta per grandi dataset.

Contro:

- Instabile nell'ottimizzazione, sensibile alla variazione dei singoli campioni.
- Può necessitare di molte iterazioni per convergere.
- Rischia overfitting, specialmente con dataset rumorosi o non rappresentativi.



Stochastic Gradient Descent

Ottimizzatori

GD, SGD e Mini-Batch SGD: pro e contro

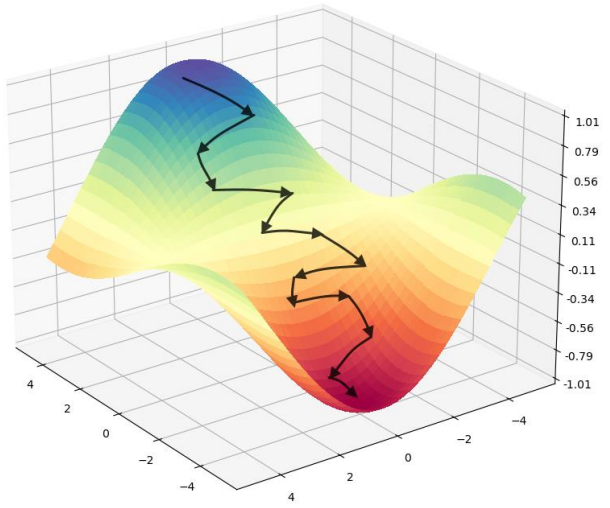
Mini-Batch SGD:

Pro:

- Più stabile nell'ottimizzazione rispetto a SGD.
- Più efficiente grazie all'utilizzo di mini-batch.
- Scalabile e adattabile a una vasta gamma di problemi.

Contro:

- Richiede la messa a punto di iperparametri come la dimensione del mini-batch.
- Può richiedere più iterazioni per convergere rispetto a Gradient Descent classico.
- Rischio di overfitting, specialmente con mini-batch troppo piccoli.



Mini-Batch Gradient Descent

Ottimizzatori

SGD con momentum

Algoritmi che sfruttano il ***Momentum*** (β) si basano sul concetto di '*velocità*'.

Nella pratica, il Momentum è un fattore moltiplicativo che:

- Mette in relazione il gradiente corrente con i gradienti passati.
- Permette di bilanciare quanto il *passato* conta rispetto al *presente*.

Intuitivamente, la domanda che ci si deve porre è:

*Un errore **grave** successivo ad una storia di piccoli errori, quanto conta?*

Un piccolo errore dopo una storia di gravi errori, quanto conta?

Ottimizzatori

SGD con momentum

Sfruttando i gradienti passati è possibile far perdurare la direzione presa da una modifica:

- Aiutando ad evitare minimi locali.
- Accelerando la convergenza.

SGD

$$W_{new} = W_{old} - \alpha * \frac{\partial(Loss)}{\partial(W_{old})}$$

SGD con momentum

$$W_{new} = W_{old} - \alpha * V_{new}$$

$$V_{new} = \beta V_{old} - (1 - \beta) * \frac{\partial(Loss)}{\partial(W_{old})}$$

Ottimizzatori

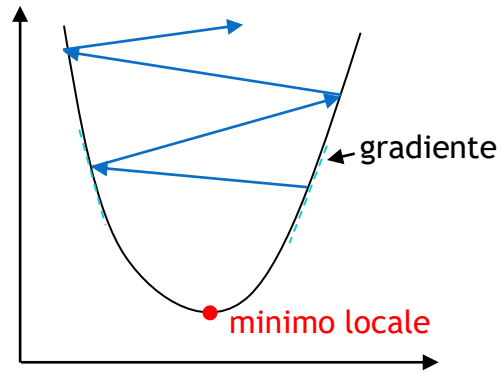
Exploding e Vanishing Gradient

Nel caso di funzioni complesse come le reti neurali, i gradienti possono tendere ad aumentare esponenzialmente o ridursi fino ad annullarsi, man mano che i dati attraversano la rete.

Questi due problemi vengono rispettivamente chiamati:

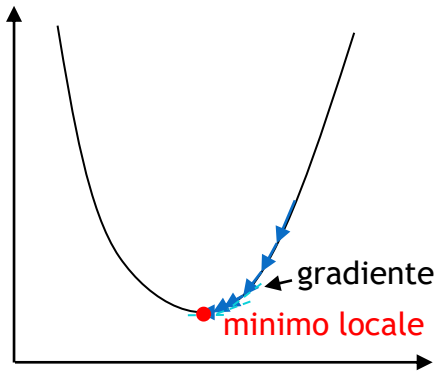
Exploding gradient

I gradienti aumentano fino a portare divergenza durante la fase di addestramento.



Vanishing gradient

I gradienti diminuiscono tendendo allo zero fino a che i pesi non vengono più aggiornati e il modello non apprende più.



Ottimizzatori

Learning rate: da fisso ad adattivo

Cercare di ottimizzare una funzione di costo mantenendo **fisso** il valore del tasso di apprendimento *(learning rate)* può comportare diversi svantaggi:

- *Convergenza lenta*.
- *Oscillazioni*.
- *Stop in minimi o selle*.
- *Poca adattabilità a variazioni dei campioni*.

Alcuni esempi possono riassumere la maggior parte di questi concetti.

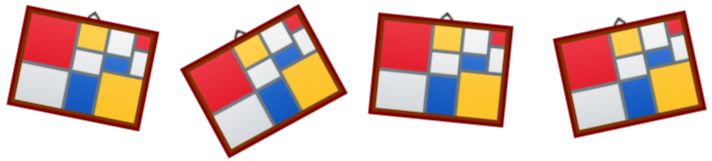
Ottimizzatori

Learning rate: da fisso ad adattivo

Immagina di prendere un libro chiuso e voler raggiungere pagina *542*…

…girando una pagina alla volta, avanti e indietro.
…o girando grandi blocchi di pagine.

Immagina di voler raddrizzare un quadro storto…
…ma ogni volta esagerare con la correzione.



Immagina di dover caricare di pacchi il retro di un camion…

…lanciando i pacchi sul retro, fragili o non fragili che siano.

…o poggiando ogni singola scatola con delicatezza, anche se non sarebbe necessario.

Ottimizzatori

Apprendimento adattivo

Nuovi ottimizzatori nascono quindi con lo scopo di:

1. Correggere in maniera dinamica rispetto lo stato dell'addestramento in cui ci si trova
2. Trattare ogni parametro in una maniera più consona alla sua 'storia di errori.

AdaGrad

- Scala il learning rate per ogni parametro della rete.
- Accumula il quadrato dei gradienti di ogni parametro e aggiorna con un learning rate rapportato a questo accumulatore.
- Accumulare il gradiente e rapportarlo al learning rate può portare però ad avere un tasso di apprendimento sempre più basso ottenendo quindi un addestramento adattivo ma lento o inefficace.

RMSProp

- Utilizza una media mobile esponenziale dei quadrati dei gradienti anziché la somma dei quadrati.
- Questa media segue la storia dei gradienti pesando in maniera sempre maggiore i più 'recenti' rispetto i meno recenti.
- Mitiga il problema della drastica riduzione del tasso di apprendimento ottenuta da *AdaGrad*.

Ottimizzatori

Apprendimento adattivo

Nuovi ottimizzatori nascono quindi con lo scopo di:

1. Correggere in maniera dinamica rispetto lo stato dell'addestramento in cui ci si trova
2. Trattare ogni parametro in una maniera più consona alla sua 'storia di errori.

ADAM

- Utilizza una media mobile esponenziale dei quadrati dei gradienti.
- Utilizza una media mobile esponenziale dei gradienti.
- Sfrutta la prima di queste medie per adattare il tasso di apprendimento alla *scala* degli errori commessi da ogni parametro.
- Sfrutta la seconda di queste medie per adattare il tasso di apprendimento alla *direzione* nella quale sono stati commessi gli errori.

AdaDelta

- Variante di *ADAM* che elimina la necessità di specificare un learning rate iniziale.
- Cerca di riadattare l'apprendimento solamente sulla storia dei gradienti.

Ottimizzatori

Oprimizers in PyTorch

PyTorch mette a disposizione diversi algoritmi di ottimizzazione tramite il modulo dedicato *torch.optim:*

Algoritmi di ottimizzazione in PyTorch

Di seguito i riferimenti specifici:

- *SGD* : *SGD*
- *AdaGrad* : *AdaGrad*
- *RMSProp* : *RMSProp*
- *Adam* : *Adam*
- *AdaDelta* : *AdaDelta*

Proviamo?