Proprietà di convergenza ad un punto stazionario del gradient descent.

Il Gradient Descent aggiorna i parametri con

$$w_{k+1} = w_k - \eta \nabla C(w_k)$$

w è il vettore dei parametri e η è il learning rate.

La discesa del gradiente trova il minimo globale se la funzione costo *C* è convessa e se sono soddisfatte le seguenti condizioni:

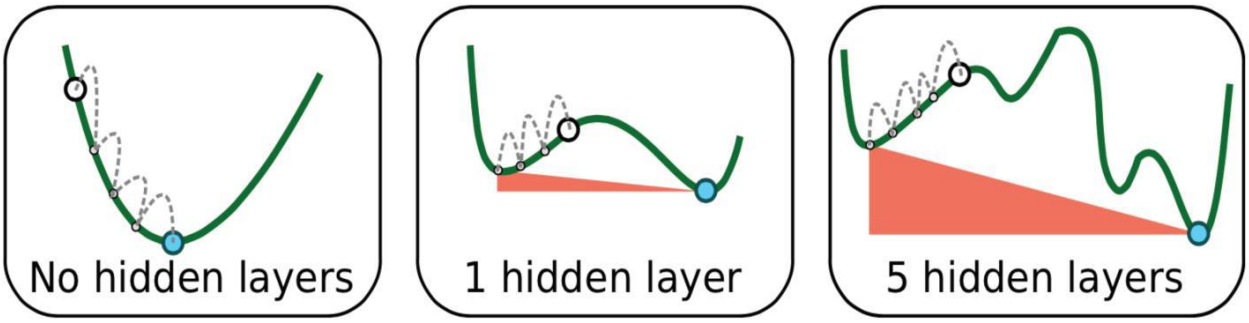
Condizione di Lipschitz per il gradiente: La norma del gradiente della funzione costo deve essere limitata da una costante di Lipschitz. Cioè , esiste una costante L > 0 tale che per ogni vettore di pesi x ed y, si abbia $\|\nabla C(x) - \nabla C(y)\| \leq L \|x - y\|$, dove $\nabla C(x)$ rappresenta il gradiente della funzione obiettivo f in x. Questa condizione impone che il gradiente non cresca troppo velocemente, e che la funzione costo sia sufficientemente regolare.

Step-size (o learning rate) : Il passo di aggiornamento η deve essere scelto in base alla costante di Lipschitz L. In particolare, per garantire la convergenza, il passo η deve essere minore o uguale a 1/L. Questa condizione assicura che il passo non sia troppo grande rispetto alla crescita del gradiente, limitando così la possibilità di oscillazioni e garantendo la convergenza del metodo.

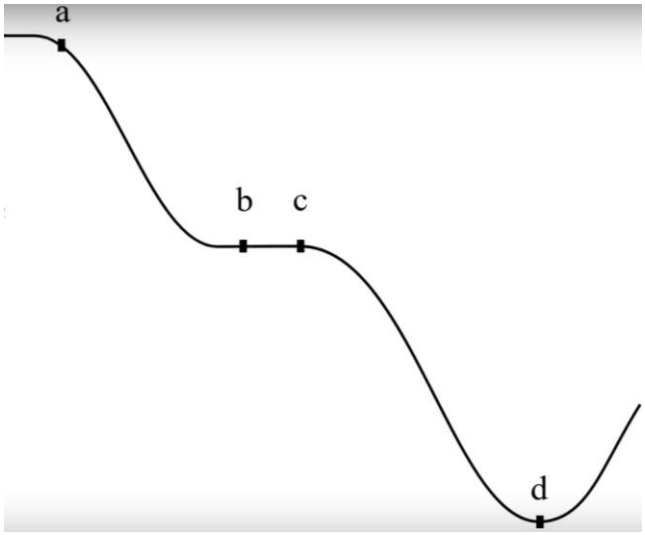
Sotto queste condizioni, il metodo di discesa del gradiente con passo fisso converge a un punto stazionario della funzione costo, che può essere un minimo globale se la funzione è convessa. Tuttavia, è importante sottolineare che la scelta del passo η fisso può essere limitante in termini di efficienza e velocità di convergenza in problemi più complessi**. In pratica, spesso si utilizzano metodi di discesa del gradiente con passo adattivo**, come ad esempio il metodo di discesa del gradiente con regola di Armijo o altre strategie avanzate, per ottenere una migliore convergenza.

Sfortunatamente, quasi tutti i problemi di ottimizzazione che sorgono in Deep Learning sono non convessi

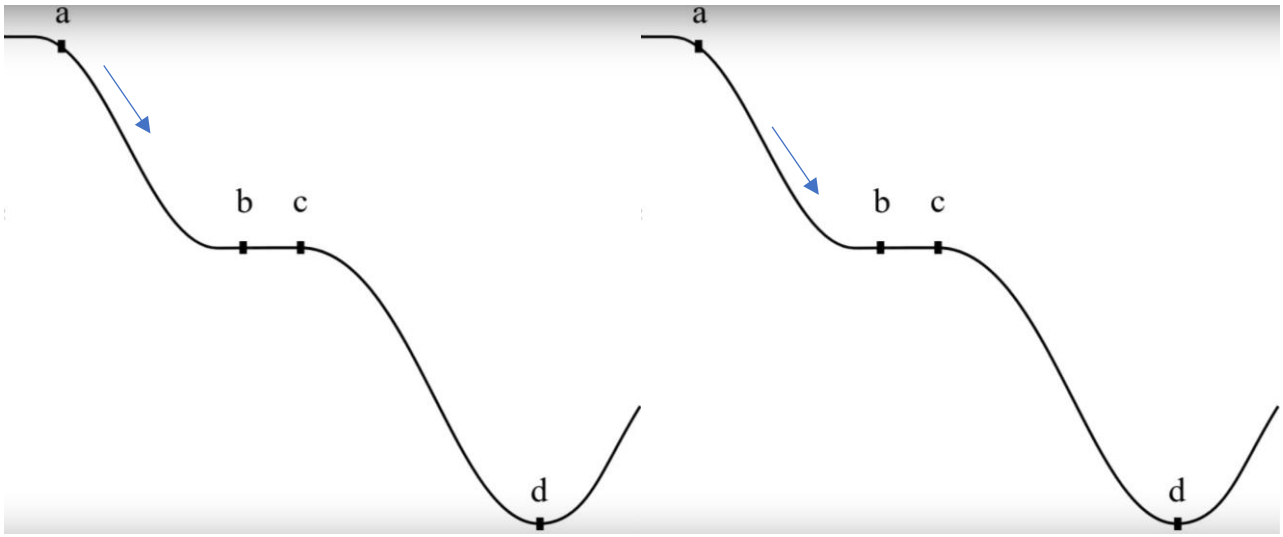
• introducendo la non linearità nella rete (aggiungendo strati nascosti), la funzione costo diventa **non convessa e compaiono i minimi locali**

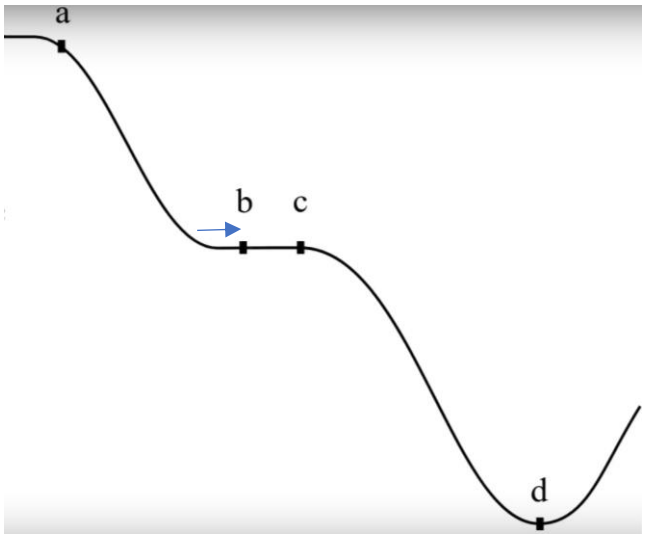


Consideriamo questa curva

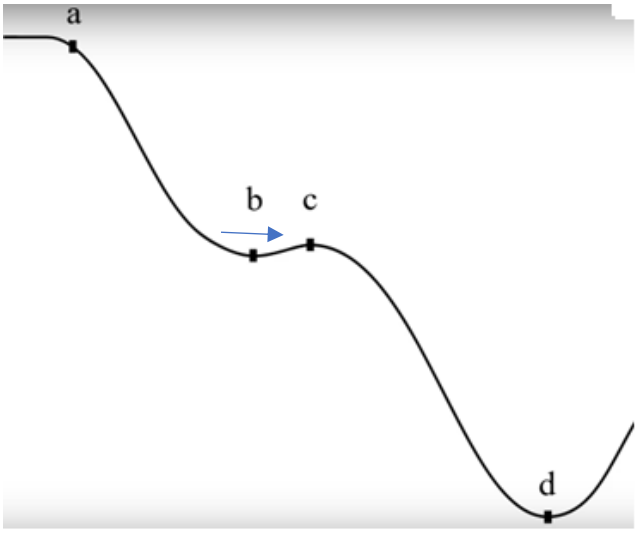


Inizializziamo w ad un punto a ed applichiamo il metodo iterativo del gradiente.





w raggiunge il punto b e si ferma, poiché in b il gradiente ai annulla, la funzione è caratterizzata da una zona piatta, un $plateau$ (*vanishing gradient*). Se invece di una regione piatta, in b c'è un minimo locale



w non può superare il massimo locale nel punto c e rimane bloccato in b.

In entrambi i casi non può raggiungere il minimo locale in d.

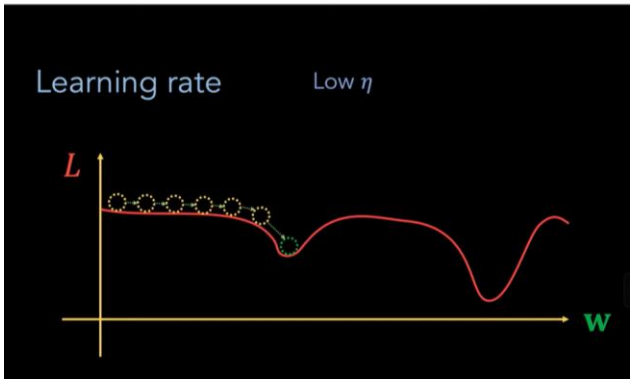
Simili situazioni si possono verificare nelle superfici multidimensionali. Specialmente nei punti sella dove il gradiente è zero, ma il punto non è un minimo oppure un massimo.

Importanza del Learning rate

Valori bassi di η

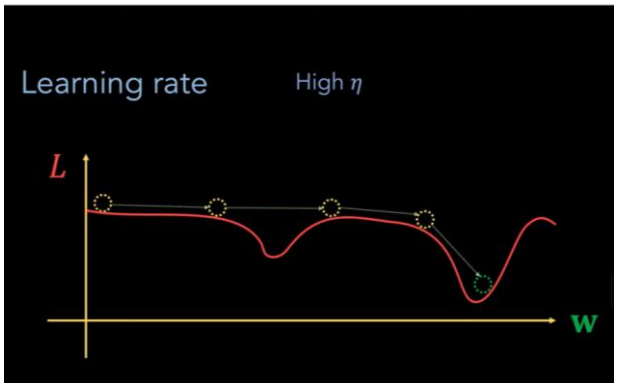
- possono far sì che sia necessario un numero elevato di passi prima che l'allenamento sia completato.

- possono far sì che i pesi rimangano bloccati in un minimo locale, che è sub-optimal se confrontato con il global minimum.

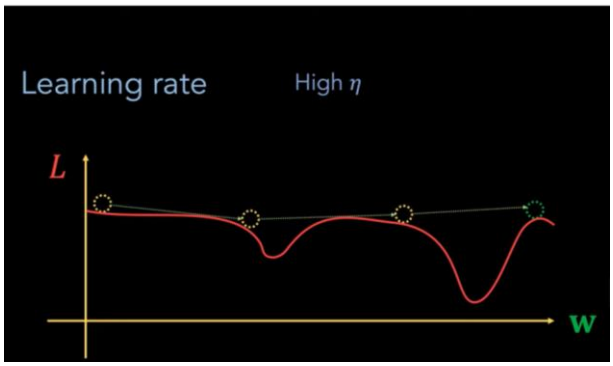


Valori alti di η

Valori alti di learning rate permettono di mitigare questi problemi passando oltre i minimi locali e raggiungendo il minimo in numero limitato di passi.



i pesi possono però anche finire per superare il minimo target



Queste osservazioni fanno intuire che la scelta del learning late è cruciale ed in pratica può richiedere molta sperimentazione per scoprire quale può essere il valore ottimale del learning rate.

Exact line search

La dimensione ottimale del learning rate può essere trovata risolvendo un problema di ottimizzazione unidimensionale

$$\eta^{(k)} = \arg\min_{\eta \geq 0} C(w^{(k)} + \eta p^{(k)})$$

dove $p^{(k)}$ è la direzione di discesa. Gli algoritmi di ottimizzazione unidimensionale per trovare la dimensione del passo ottimale sono genericamente chiamati **exact line search**

Sono state proposte delle **inexact line search rule .** Tra le quali la prima è stata

Armijo rule

la regola di Armijo consente di trovare uno step-size η appropriato che garantisca una riduzione significativa della funzione costo, garantendo al contempo un'adeguata convergenza dell'algoritmo di ottimizzazione,

La regola di Armijo, chiamata anche regola di Armijo-Goldstein, è un criterio utilizzato nell'ottimizzazione numerica per determinare la dimensione del passo o lo step size da utilizzare durante la ricerca lineare in direzione di discesa all'interno di un algoritmo di ottimizzazione.

La regola di Armijo si basa sul concetto di **"backtracking",** che prevede di ridurre gradualmente la dimensione del passo fino a trovare un valore che soddisfi determinate condizioni. In particolare, la regola di Armijo si concentra sulla scelta di un passo che garantisca una riduzione sufficiente del valore della funzione costo durante la ricerca lineare.

L'idea principale è che, partendo da un certo punto iniziale, si riduca progressivamente la dimensione del passo moltiplicandola per il parametro σ ($0 < \sigma < 1$) fino a quando non si ottiene una riduzione sufficiente nella funzione obiettivo. Questo processo viene ripetuto fino a quando la condizione di Armijo è soddisfatta .

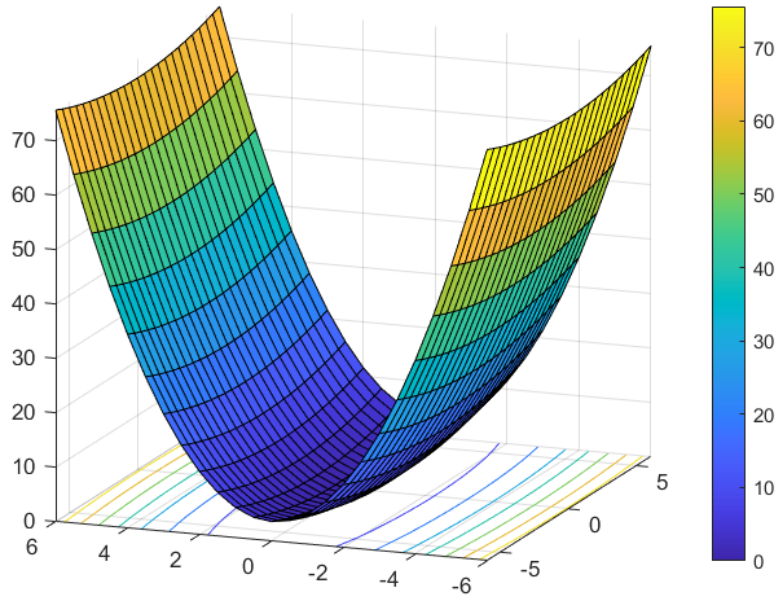
Se questa condizione viene soddisfatta

$$C(w^{(k)} + \eta p^{(k)}) \leq C\left(w^{(k)}\right) + \sigma \cdot \eta \nabla C\left(w^{(k)}\right)^T p^{(k)}$$

allora il passo η viene accettato; altrimenti, il passo viene ridotto di un fattore σ e il processo viene ripetuto.

Armijo rule: parte con $\eta = \eta_0$ e lo fa decrescere moltiplicandolo per $\sigma \in (0,1)$, finchè la funzione descresce sufficientemente.

Consideriamo adesso una lossy del tipo $f(w_1, w_2) = 0.1\ w_1^2 + 2\ w_2^2$



Ha il suo minimo in (0,0) . Questa funzione è *molto* piatta nella direzione di w_1.

$$\nabla f(w_1, w_2) = \begin{bmatrix} \frac{\partial f(w_1, w_2)}{\partial w_1} \\ \frac{\partial f(w_1, w_2)}{\partial w_2} \end{bmatrix} = \begin{bmatrix} 0.2\ w_1 \\ 4 w_2 \end{bmatrix}$$

Analizzando la prima componente del gradiente, la derivata parziale rispetto a w_1, notiamo che varia poco $(0.2\ w_1)$ rispetto alla seconda componente, , la derivata parziale rispetto a w_2, $(4\ w_2)$

Vediamo cosa succede quando eseguiamo la discesa del gradiente per calcolare il minimo della funzione f. Scegliamo un learning rate $\eta = 0.4$.

$\boldsymbol{w_1^{(0)} = -5,\ w_2^{(0)} = -2,}$

$f\left(w_1^{(0)}, w_2^{(0)}\right) = -10.5$

$w_1^{(1)} = w_1^{(1)} - \eta \frac{\partial f}{\partial w_1}\left(w_1^{(0)}, w_2^{(0)}\right) =$

$= -5 - 0.4(0.2 \cdot -5) = -4.6$

$w_2^{(1)} = w_2^{(1)} - \eta \frac{\partial f}{\partial w_2}\left(w_1^{(0)}, w_2^{(0)}\right) =$

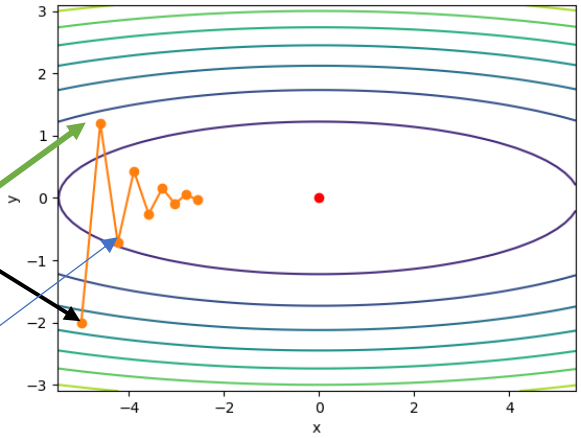
$-2 - 0.4\left(4 \cdot (-2)\right) = 1.2$

$\boldsymbol{w_1^{(1)} = -4.6,\ w_2^{(1)} = 1.2,}$

$f\left(w_1^{(1)}, w_2^{(1)}\right) = 5$

$\boldsymbol{w_1^{(2)} = -4.2,\ w_2^{(2)} = -0.7,}$

$f\left(w_1^{(2)}, w_2^{(2)}\right) = 2.7$



Per come è definita la funzione costo $f(w_1, w_2)$ il gradiente nella direzione w_2 è *molto* più alto e cambia molto più rapidamente rispetto alla direzione orizzontale. Quindi siamo bloccati tra due scelte: **se scegliamo un learning rate basso che la soluzione non diverge nella direzione verticale w_2 , ma siamo gravati da una lenta convergenza nella direzione orizzontale w_1. Viceversa, con un elevato tasso di apprendimento progrediamo rapidamente nella direzione orizzontale, ma c'è divergenza nella direzione verticale.**

`Dopo 20 epoche` $w_1 : -0.943467, w_2 : -0.000073$

Per risolvere questi problemi si introduce il gradient descent con momento.

Il **Gradient Descent con momento** viene utilizzato per risolvere i problemi elencati.

Si definisce la **velocità** $v_k = \beta v_{k-1} + \nabla C(w_k)$

β prende il nome di momento e può variare tra 0 ed 1. v_0 viene inizializzato a zero.

La formula di aggiornamento dei pesi diventa:

$$w_{k+1} = w_k - \eta v_k$$

Se si pone $\beta = 0$, si ricade nel classico metodo del gradiente. Un valore appropriato è tra 0.8 e 0.9.

Questo metodo aggiunge al gradiente istantaneo la media pesata esponenzialmente su più gradienti *passati* . Di seguito verifichiamo che v_k rappresenta la media pesata esponenzialmente dei gradienti passati fino a quello al passo k.

Analizziamo il termine $v_k = \beta v_{k-1} + \nabla C(w_k)$

$v_1 = \beta v_0 + \nabla C(w_1)$

$v_2 = \beta v_1 + \nabla C(w_2)$

$v_2 = \beta\big(\beta v_0 + \nabla C(w_1)\big) + \nabla C(w_2) = \beta^2 v_0 + \beta \nabla C(w_1) + \nabla C(w_2)$

$$v_3 = \beta v_2 + \nabla C(w_3) = \beta\big(\beta^2 v_0 + \beta \nabla C(w_1) + \nabla C(w_2)\big) + \nabla C(w_3)$$
$$= \beta^3 v_0 + \beta^2 \nabla C(w_1) + \beta \nabla C(w_2) + \nabla C(w_3)$$

$$v_k = \beta^k v_0 + \beta^{k-1} \nabla C(w_1) \ldots + \beta \nabla C(w_{k-1}) + \beta^0 \nabla C(w_k)$$

Posto $v_0 = 0$

$$v_k = \sum_{j=1}^{k} \beta^{k-j} \nabla C(w_j)$$

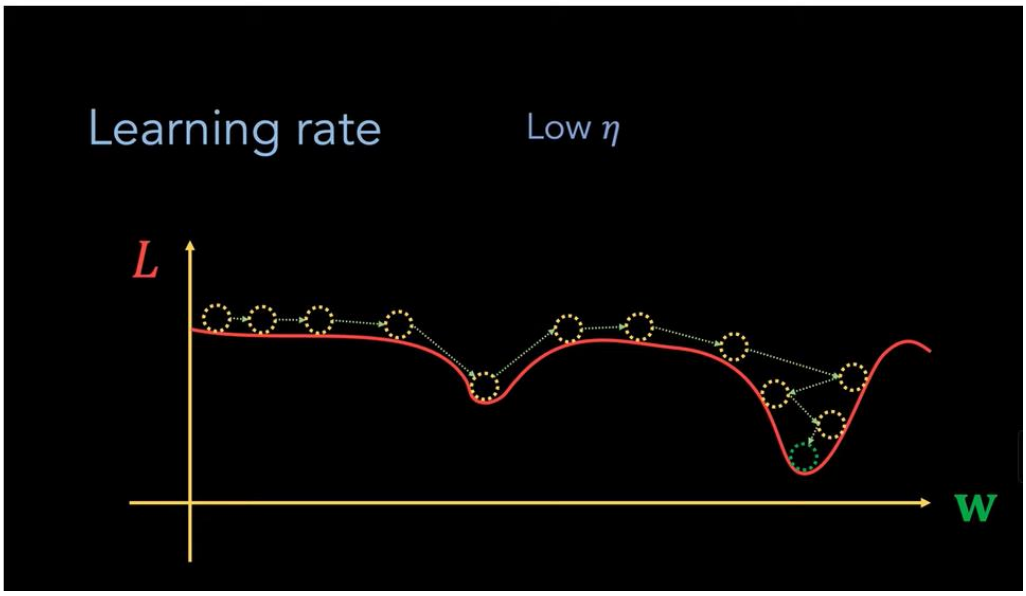
Si basa sullo stesso concetto di **momento** , *quantità di moto in fisica*. Un classico esempio del concetto è una palla che rotola giù da una collina che raccoglie abbastanza slancio per superare una regione di altopiano e raggiungere un minimo globale invece di rimanere bloccata in un minimo locale.

Il metodo SGD con Momento aggiunge la "cronologia" agli aggiornamenti dei parametri dei problemi di discesa, il che accelera notevolmente il processo di ottimizzazione.

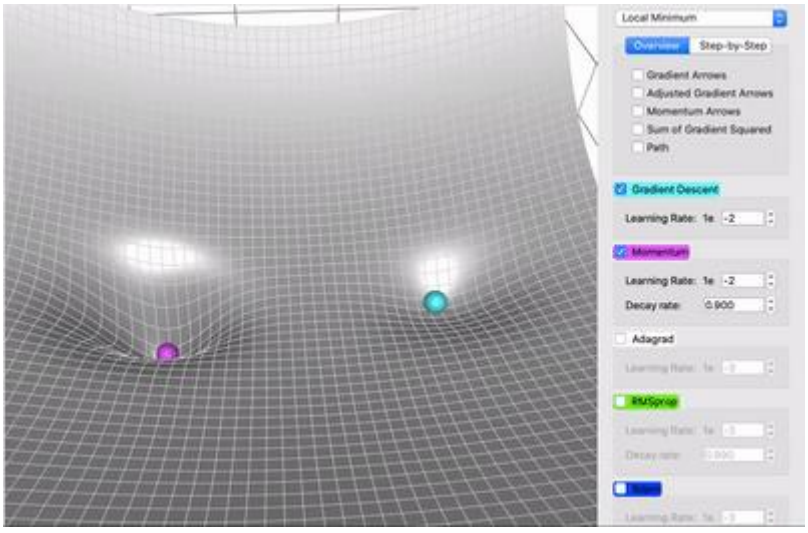
La nuova sostituzione del gradiente non punta più nella direzione della discesa più ripida in un caso particolare, ma piuttosto nella direzione di una media ponderata esponenziale dei gradienti passati.

Il termine **momento aumenta le dimensioni degli aggiornamenti quando i gradienti puntano nella stessa direzione** mentre **riduce le dimensioni degli aggiornamenti quando i gradienti cambiano direzione** (gradienti oscillanti hanno somma piccola in quanto le oscillazioni tra valori positivi e valori negativi del gradiente portano ad avere una media più piccola)
In questo modo si ottiene una più veloce convergenza e si riduce l'oscillazione.

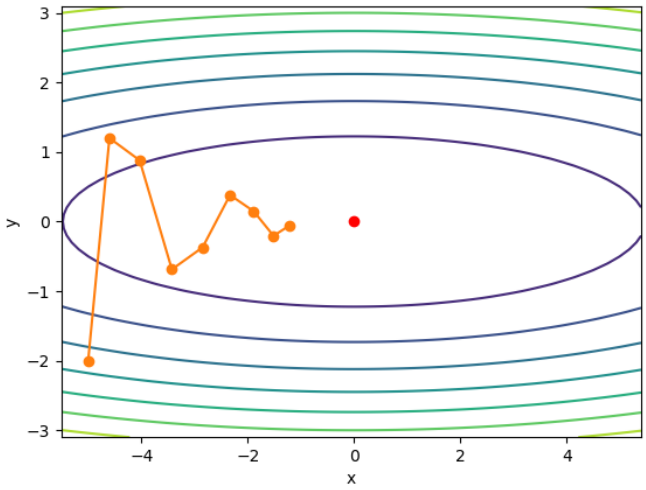
In questo modo la scelta del learning rate è meno cruciale poiché la procedura di training si adatta alla particolare loss function.



La quantità di cronologia inclusa nell'equazione di aggiornamento è determinata tramite l' iperparametro β Questo iperparametro è un valore compreso tra 0 e 1, dove un valore di quantità di moto pari a 0 equivale alla discesa del gradiente senza quantità di moto Un valore di momentum più alto significa che vengono considerati più gradienti del passato (storia).



Tornando al nostro esempio, il metodo del gradiente con momento produrrà le seguenti iterazioni:



nella direzione w_1 il metodo del gradiente con momento accumulerà gradienti ben allineati, aumentando così la distanza che percorriamo ad ogni passo. Al contrario, nella direzione $w_2$in cui oscillano i gradienti, l'accumulo dei gradienti ridurrà la dimensione del passo a causa delle oscillazioni che si annullano a vicenda.

Il ragionamento di cui sopra ha costituito la base per quelli che oggi sono noti come metodi del gradiente *accelerato* , come i gradienti con momento (quantità di moto) . Godono dell'ulteriore vantaggio di essere molto più efficaci nei casi in cui il problema di ottimizzazione è mal condizionato (cioè, dove ci sono alcune direzioni in cui il progresso è molto più lento che in altre, assomigliando a uno stretto canyon). Inoltre, ci permettono di fare la media sulle pendenze successive per ottenere direzioni di discesa più stabili. In effetti, l'aspetto dell'accelerazione anche per problemi convessi senza rumore è uno dei motivi principali per cui la quantità di moto funziona e perché funziona così bene. http://distill.pub/2017/momentum , doi:10.23915/distill.00006

Tuttavia, una palla che rotola giù da una collina, seguendo ciecamente il pendio, è altamente insoddisfacente

Gradiente accelerato di Nesterov usa un termine di momento più intelligente, che ha un'idea di dove sta andando in modo che sappia rallentare prima che la funzione salga di nuovo.