# Progetto di Analisi Numerica Anno 2024–2025 (LM-18 Informatica)

Confronto con Metodi Iterativi e Applicazione ad un Modello di Machine-Learning
per il Learning dei Parametri

Problema: Convergenza del Minimo di funzioni NON LINEARI (Rosenbrock)

Nome Cognome Emanuele Valore
Matricola: 1000030355

25 maggio 2025

# Indice


**1 Introduzione** **3**
- 1.1 Funzione di Rosenbrock . . . 3
- 1.2 Agganci Teorici: . . . 3
  - 1.2.1 Teorema del Gradiente e dei Minimi Locali . . . 3
  - 1.2.2 Il Teorema di Fermat . . . 3
  - 1.2.3 Teorema del Secondo Ordine (Hessiano) . . . 4

**2 Stochastic Gradient Descent** **5**
- 2.1 Introduzione Al Metodo: . . . 5
  - 2.1.1 Formula Generale . . . 5
- 2.2 Tipologie di Discesa del Gradiente . . . 5
  - 2.2.1 Batch Gradient Descent (BGD) . . . 5
  - 2.2.2 Stochastic Gradient Descent (SGD) . . . 6
  - 2.2.3 Mini-Batch Gradient Descent (MBGD) . . . 6
- 2.3 Tasso di Apprendimento (Learning Rate) . . . 6
- 2.4 Ottimizzazioni di SGD . . . 6
  - 2.4.1 Momentum . . . 7
  - 2.4.2 Learning Rate Scheduling . . . 7
  - 2.4.3 Mini-Batch Gradient Descent . . . 7

**3 Metodi Hessiani** **8**
- 3.1 Introduzione al Metodo . . . 8
- 3.2 Ottimizzazione Basata sull'Hessiano . . . 8
- 3.3 Il Metodo BFGS . . . 8
  - 3.3.1 Vantaggi del Metodo BFGS . . . 9
  - 3.3.2 Implementazione Pratica del BFGS . . . 9

**4 Applicazione a RosenBrock e Risultati** **10**
- 4.1 Derivazione della Funzione di Rosenbrock . . . 10
- 4.2 Spiegazione dei Valori Trovati per Gradiente ed Hessiano . . . 10
  - 4.2.1 Nel Caso della Funzione di Rosenbrock . . . 10
- 4.3 Plot della Funzione di Rosenbrock in 3D . . . 10
- 4.4 Interpretazione dei Risultati (SGD Adam . . . 12
  - 4.4.1 Tabella dei Risultati . . . 12
  - 4.4.2 Interpretazione Finale dei Metodi SGD-ADAM . . . 12
- 4.5 Interpretazione Finale dei Metodi HESSIANO-BFGS . . . 13
- 4.6 Tabella dei Risultati . . . 13

**5 Applicazione Pratica** **14**
5.1 Applicazione Nel Machine Learning: . . . . . 14
5.1.1 Visualizzazione della Funzione di Rosenbrock . . . . . 14
5.2 Conclusioni . . . . . 14

# Capitolo 1

# Introduzione

In questa relazione si affronta il problema della *convergenza del minimo* di alcune funzioni non lineari, con particolare attenzione alla funzione di Rosenbrock. L'obiettivo è confrontare metodi iterativi di ottimizzazione (*ad esempio: discesa del gradiente, metodi quasi-Newton, metodi hessiani*) e applicarli ad un semplice modello di *Machine Learning* (ad esempio un *Multilayer Perceptron*) per il *learning* dei parametri.

## 1.1 Funzione di Rosenbrock

La funzione di Rosenbrock è definita come:

$$f(x, y) = (a - x)^2 + b(y - x^2)^2,$$

dove di solito si imposta $a = 1$ e $b = 100$. Questa funzione ha un minimo globale in $(x, y) = (1, 1)$ ma presenta una valle stretta e curva che rende difficile la convergenza di molti metodi iterativi quando il punto di partenza è lontano dal minimo.Questa funzione viene utilizzata per valutare i metodi iterativi in particolare la loro convergenza dato che essa presenta una stretta curva di minimo.

## 1.2 Agganci Teorici:

### 1.2.1 Teorema del Gradiente e dei Minimi Locali

Il gradiente di una funzione in un punto fornisce la direzione di massima crescita della funzione. Se il gradiente della funzione è zero in un punto, questo è indicativo che tale punto possa essere un minimo, un massimo o un punto di sella.

### 1.2.2 Il Teorema di Fermat

Il Teorema di Fermat stabilisce che, se una funzione differenziabile ha un massimo o un minimo locale in un punto, allora il gradiente della funzione in quel punto è zero. Questo è esattamente il caso osservato: il gradiente della funzione di Rosenbrock in $(1.0, 1.0)$ è zero, suggerendo che $(1.0, 1.0)$ sia un punto critico. La verifica dell'Hessiano ha permesso di determinare che si tratta di un minimo locale.

### 1.2.3 Teorema del Secondo Ordine (Hessiano)

Il teorema del secondo ordine stabilisce che la natura del punto critico (minimo, massimo o punto di sella) dipende dalla matrice Hessiana. In particolare:

- Se la matrice Hessiana è **definita positiva** (tutti i suoi autovalori sono positivi), allora il punto è un **minimo locale**.
- Se la matrice Hessiana è **definita negativa** (tutti i suoi autovalori sono negativi), allora il punto è un **massimo locale**.
- Se la matrice Hessiana è **indefinita** (alcuni autovalori sono positivi e altri negativi), allora il punto è un **punto di sella**.

# Capitolo 2

# Stochastic Gradient Descent

## 2.1 Introduzione Al Metodo:

Il metodo della discesa del gradiente è un algoritmo di ottimizzazione iteratico utilizzato per trovare il minimo di una funzione matematica, in particolare viene utilizzato nel Machine Learning per andare a minimizzare una funzione di Costo, che è la funzione di Costo (indicativa del numero di errori effettuati dal modello di ML). L'idea di Base è quella di muoversi nella direzione del gradiente negativo della funzione di costo per raggiungere il minimo Locale e globale. Il gradiente indica la direzione di massima crescita della funzione, muovendosi in direzione opposta andiamo verso il minimo.

### 2.1.1 Formula Generale

Dato un insieme di parametri $\theta$, una funzione obiettivo $J(\theta)$ da minimizzare e un tasso di apprendimento $\alpha$, l'aggiornamento dei parametri avviene secondo la seguente regola:

$$\theta := \theta - \alpha \nabla J(\theta),$$

dove $\nabla J(\theta)$ è il gradiente della funzione di costo rispetto ai parametri.

## 2.2 Tipologie di Discesa del Gradiente

### 2.2.1 Batch Gradient Descent (BGD)

- Utilizza l'intero dataset per calcolare il gradiente.
- Converge in modo stabile ma può essere lento per dataset molto grandi.

La formula di aggiornamento è:

$$\theta := \theta - \alpha \frac{1}{m} \sum_{i=1}^{m} \nabla J(\theta),$$

dove $m$ rappresenta la dimensione dell'intero dataset.

### 2.2.2 Stochastic Gradient Descent (SGD)

- Aggiorna i parametri ad ogni iterazione utilizzando un singolo esempio.
- Più veloce, ma introduce rumore nelle iterazioni.
- Non garantisce convergenza a un minimo globale.

La formula di aggiornamento è:

$$\theta := \theta - \alpha \nabla J(\theta_i),$$

dove $\theta_i$ indica i parametri calcolati su un singolo campione (o esempio).

### 2.2.3 Mini-Batch Gradient Descent (MBGD)

- Compromesso tra BGD e SGD.
- Utilizza piccoli *batch* del dataset per calcolare il gradiente.
- Più stabile di SGD e più efficiente di BGD.

La formula di aggiornamento è:

$$\theta := \theta - \alpha \frac{1}{B} \sum_{i=1}^{B} \nabla J(\theta_i),$$

dove $B$ è la dimensione del mini-batch.

## 2.3 Tasso di Apprendimento (Learning Rate)

Il parametro $\alpha$ controlla la grandezza del passo compiuto nella direzione del gradiente:

- Se $\alpha$ è troppo grande, si rischia di *saltare* il minimo e non convergere.
- Se $\alpha$ è troppo piccolo, la convergenza sarà molto lenta.

## 2.4 Ottimizzazioni di SGD

Nel contesto dell'ottimizzazione tramite Stochastic Gradient Descent (SGD) sono state sviluppate diverse tecniche per migliorare sia la convergenza che la stabilità del metodo. In questa sezione vengono discusse alcune delle ottimizzazioni più comuni, quali il momentum, il learning rate scheduling e l'impiego del mini-batch.

### 2.4.1 Momentum

L'aggiunta del termine di momentum mira ad accelerare la convergenza aggregando le informazioni dei gradienti precedenti. Ciò permette di ridurre la varianza degli aggiornamenti e di superare eventuali oscillazioni dovute alla natura stocastica dei campioni. La formula per l'aggiornamento con momentum è:

$$v_t = \beta v_{t-1} + (1 - \beta)\, \nabla J(\theta_t),$$

$$\theta_{t+1} = \theta_t - \alpha\, v_t,$$

dove:

- $v_t$ rappresenta il vettore di momentum al tempo $t$,
- $\beta$ è il coefficiente di momentum (tipicamente compreso tra 0.9 e 0.99),
- $\alpha$ è il tasso di apprendimento.

### 2.4.2 Learning Rate Scheduling

La regolazione dinamica del tasso di apprendimento consente di partire con un valore elevato per spostarsi rapidamente nella direzione del minimo, per poi ridurlo gradualmente per affinare la convergenza. Alcuni approcci comuni sono:

- **Step Decay:** il tasso di apprendimento viene ridotto di un fattore costante dopo un numero prefissato di epoche.
- **Exponential Decay:** il tasso di apprendimento diminuisce in modo esponenziale nel tempo.
- **Adaptive Methods:** algoritmi come AdaGrad, RMSProp e Adam adattano il tasso di apprendimento per ogni parametro in base alla storia dei gradienti.

### 2.4.3 Mini-Batch Gradient Descent

Invece di aggiornare i parametri utilizzando un singolo esempio (SGD puro) o l'intero dataset (Batch Gradient Descent), il Mini-Batch Gradient Descent suddivide il dataset in piccoli gruppi (mini-batch). Questa strategia offre vari vantaggi:

- Riduce la varianza degli aggiornamenti, migliorando la stabilità.
- Sfrutta l'ottimizzazione tramite parallelismo (specialmente con GPU).
- Bilancia l'accuratezza della stima del gradiente e l'efficienza computazionale.

La regola di aggiornamento diventa:

$$\theta_{t+1} = \theta_t - \alpha\, \frac{1}{B} \sum_{i=1}^{B} \nabla J(\theta; x_i),$$

dove $B$ rappresenta la dimensione del mini-batch e $x_i$ sono i campioni contenuti nel batch.

# Capitolo 3

# Metodi Hessiani

## 3.1 Introduzione al Metodo

In questo capitolo si affronta l'ottimizzazione basata su informazioni di secondo ordine, sfruttando l'Hessiano della funzione di costo. L'approccio tradizionale del metodo di Newton prevede l'uso diretto della matrice Hessiana per aggiornare i parametri:

$$\theta_{t+1} = \theta_t - H^{-1}(\theta_t)\nabla J(\theta_t),$$

dove $H(\theta_t)$ rappresenta la matrice delle derivate seconde della funzione $J(\theta_t)$. Tuttavia, il calcolo diretto e l'inversione dell'Hessiano risultano spesso computazionalmente onerosi e numericamente instabili, soprattutto in presenza di un elevato numero di parametri.

Per ovviare a questi problemi si adottano metodi di approssimazione e algoritmi quasi-Newton che stimano l'inverso dell'Hessiano senza doverlo calcolare esplicitamente. Tra questi, il metodo BFGS (Broyden–Fletcher–Goldfarb–Shanno) si è dimostrato particolarmente efficace.

## 3.2 Ottimizzazione Basata sull'Hessiano

I metodi hessiani utilizzano la curvatura locale della funzione di costo per determinare la direzione e l'ampiezza degli aggiornamenti dei parametri. L'idea è quella di migliorare il passo di aggiornamento sfruttando la seconda derivata:

$$\theta_{t+1} = \theta_t - H^{-1}(\theta_t)\nabla J(\theta_t).$$

Tuttavia, a causa dei costi computazionali e delle problematiche di stabilità, si preferiscono metodi che approssimano $H^{-1}(\theta_t)$.

## 3.3 Il Metodo BFGS

Il metodo BFGS è un algoritmo quasi-Newton che aggiorna iterativamente una matrice $B_k$ che approssima l'inverso dell'Hessiano. L'aggiornamento di $B_k$ avviene secondo la seguente formula:

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k},$$

dove:

- $s_k = \theta_{k+1} - \theta_k$ è il cambiamento nei parametri,
- $y_k = \nabla J(\theta_{k+1}) - \nabla J(\theta_k)$ è il cambiamento nel gradiente.

Questa formula permette di aggiornare la stima dell'inverso dell'Hessiano in maniera efficiente, evitando il calcolo diretto di $H(\theta_t)$ e la sua inversione.

### 3.3.1 Vantaggi del Metodo BFGS

- **Efficienza Computazionale:** Il metodo evita il calcolo esplicito dell'Hessiano, riducendo il costo computazionale.
- **Convergenza Rapida:** In molti problemi, BFGS converge più rapidamente rispetto ai metodi basati solo sul gradiente.
- **Stabilità Numerica:** L'aggiornamento iterativo della matrice $B_k$ garantisce una buona approssimazione dell'inverso dell'Hessiano e migliora la stabilità degli aggiornamenti.

### 3.3.2 Implementazione Pratica del BFGS

In ambito di machine learning, framework come PyTorch forniscono ottimizzatori che implementano algoritmi quasi-Newton. Ad esempio, `torch.optim.LBFGS` è un'implementazione che sfrutta un approccio simile al BFGS, particolarmente utile per modelli con un numero relativamente basso di parametri. Per modelli di dimensioni maggiori, possono essere adottate varianti o metodi che limitano il numero di aggiornamenti memorizzati, bilanciando precisione e costo computazionale.

# Capitolo 4

# Applicazione a RosenBrock e Risultati

## 4.1 Derivazione della Funzione di Rosenbrock

Il calcolo della derivata prima parziale e della derivata seconda parziale viene effettuato poiché questi valori sono utilizzati nei metodi iterativi per il calcolo del nuovo minimo della funzione.

## 4.2 Spiegazione dei Valori Trovati per Gradiente ed Hessiano

Nel punto

$$(x, y) = (1.0,\ 1.0)$$

il gradiente assume il valore

$$[0, 0],$$

un risultato atteso e coerente, in quanto in questo punto la funzione di **Rosenbrock** presenta un minimo locale. (Il gradiente in un minimo locale è nullo.)

### 4.2.1 Nel Caso della Funzione di Rosenbrock

I valori degli autovalori calcolati per la matrice Hessiana sono [800, 200]. Pertanto, la matrice risulta definita **positiva** e il punto (1.0, 1.0) è un minimo locale.

## 4.3 Plot della Funzione di Rosenbrock in 3D

Il plot della funzione in 3D è utile per visualizzare le curvature che essa assume in uno spazio tridimensionale. In particolare, questo grafico permette di osservare l'intera struttura del problema: mentre l'obiettivo è convergere verso il minimo della funzione, in presenza di multiple curvature alcuni metodi potrebbero convergere verso minimi locali anziché al minimo globale. Per questa ragione, modificheremo e confronteremo i metodi di ottimizzazione per valutare quale offra un comportamento migliore.

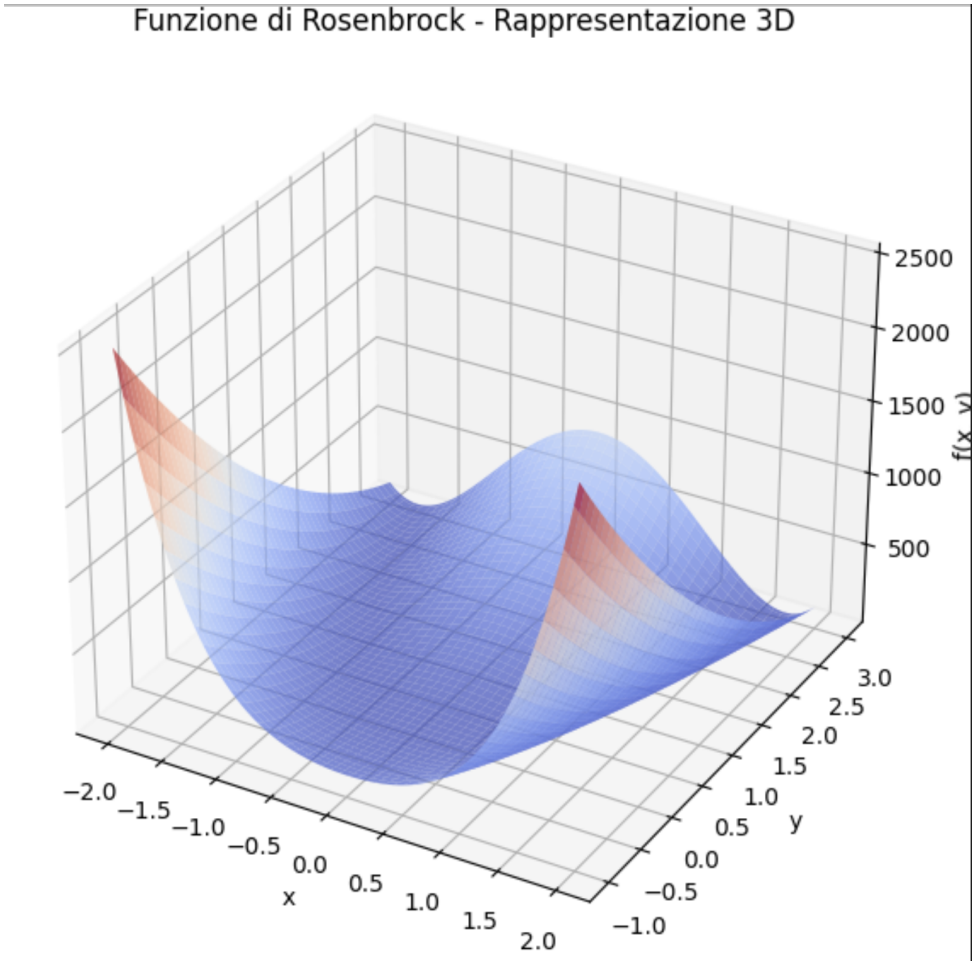


Figura 4.1: Plot RosenBrock

## 4.4 Interpretazione dei Risultati (SGD Adam

Tra i valori in output abbiamo:

- **Posizione finale:** indica dove il metodo si è fermato, ovvero il punto di minimo/-massimo trovato.
- **Valore della funzione:** ci aspettiamo che questo valore sia il più vicino possibile a quello corrispondente al minimo globale (per la funzione di Rosenbrock, attorno a $(1, 1)$).
- **Numero di iterazioni:** rappresenta il numero di aggiornamenti iterativi (calcolo dei gradienti) effettuati per cercare il minimo.

### 4.4.1 Tabella dei Risultati

Tabella 4.1: Risultati ottenuti con Discesa del Gradiente e Adagrad

| Iterazioni | Metodo | Posizione finale | Valore della funzione | Iterazioni |
|---|---|---|---|---|
| 100 | Discesa del Gradiente | (-0.96349, 0.93634) | 3.86174 | 100 |
| 100 | Adagrad | (1.78074, 0.64033) | 641.06383 | 100 |
| 1000 | Discesa del Gradiente | (0.67902, 0.45953) | 0.10326 | 1000 |
| 1000 | Adagrad | (-1.43518, 1.97190) | 6.70173 | 1000 |
| 10000 | Discesa del Gradiente | (0.99319, 0.98639) | 4.65e-05 | 10000 |
| 10000 | Adagrad | (-0.20654, -0.00534) | 1.68614 | 10000 |

### 4.4.2 Interpretazione Finale dei Metodi SGD-ADAM

**Discesa del Gradiente**

- **100 Iterazioni:** La discesa del gradiente converge verso un minimo locale, ma la posizione finale risulta lontana dal minimo globale $(1, 1)$. Il valore della funzione (3.86) è relativamente alto, indicando che il metodo necessita di ulteriori iterazioni per migliorare la convergenza.
- **1000 Iterazioni:** Con un maggior numero di iterazioni, il metodo si avvicina significativamente al minimo globale, riducendo il valore della funzione a 0.10.
- **10000 Iterazioni:** La discesa del gradiente raggiunge una posizione finale molto vicina a $(1, 1)$ e il valore della funzione diventa estremamente basso (4.65e-05), confermando una corretta convergenza al minimo globale.

**Adagrad**

- **100 Iterazioni:** Adagrad mostra una posizione finale molto distante dal minimo globale, con un valore della funzione estremamente alto (641.06). Ciò suggerisce che, a causa della rapida riduzione del tasso di apprendimento, il metodo fatica a navigare correttamente nella valle della funzione di Rosenbrock.

- **1000 Iterazioni:** Anche con un numero maggiore di iterazioni, Adagrad non migliora significativamente: la posizione finale rimane distante e il valore della funzione (6.70) risulta ancora elevato.
- **10000 Iterazioni:** Nonostante il numero elevato di iterazioni, Adagrad non riesce a convergere verso il minimo globale, mostrando una posizione finale lontana e un valore della funzione (1.69) ancora troppo alto.

## 4.5 Interpretazione Finale dei Metodi HESSIANO-BFGS

### Metodo Hessiano (100 Iterazioni)

Il metodo Hessiano, in 100 iterazioni, riesce a trovare la convergenza ottimale della funzione nel suo minimo locale. In particolare, dai risultati risulta che il metodo individua il punto $(1, 1)$, che corrisponde al minimo della funzione di Rosenbrock, e il valore della funzione risulta essere pari a 0.

### Metodo BFGS (100 Iterazioni)

Al contrario, il metodo BFGS, pur effettuando 100 iterazioni, mostra una variazione nel valore finale della funzione, dovuta al fatto che non riesce a trovare il punto ottimale in modo altrettanto preciso quanto il metodo Hessiano.

## 4.6 Tabella dei Risultati

Tabella 4.2: Risultati ottenuti con il Metodo Hessiano e il Metodo BFGS (100 Iterazioni)

| **Iterazioni** | **Metodo di Newton - Posizione finale** | **Metodo di Newton - Valore della** |
|---|---|---|
| **100 Iterazioni** | (1.0, 1.0) | 0.0 |

# Capitolo 5

# Applicazione Pratica

## 5.1 Applicazione Nel Machine Learning:

L'applicazione principale di questi algoritmi iterativi per il calcolo del minimo di una funzione differenziabile è l'ottimizzazione dei parametri di un modello di *Machine Learning*. Questo è particolarmente importante poiché i parametri del modello determinano la capacità di effettuare predizioni corrette. Nel contesto del *Machine Learning*, l'obiettivo è minimizzare la funzione di costo, che rappresenta l'errore del modello. Ridurre tale funzione significa adattare il modello nel modo migliore possibile, aumentando la precisione delle predizioni.

In questa parte del progetto, vogliamo utilizzare un dataset (reperito su *Kaggle*) per valutare come il modello di Machine Learning, da noi definito, si comporti al variare del metodo di ottimizzazione utilizzato. In particolare, seguiremo i seguenti passaggi:

- **Analisi del Dataset** trovato su Kaggle.
- **Costruzione del Modello di Machine Learning**.
- **Confronto del modello allenato con SGD e con Hessiano**.
- **Valutazioni Finali** sull'accuratezza e sulla convergenza.

### 5.1.1 Visualizzazione della Funzione di Rosenbrock

Le Figure mostrano la superficie della funzione di Rosenbrock e la posizione finale dei diversi metodi di ottimizzazione (Gradient Descent, Adagrad, Newton, BFGS).

## 5.2 Conclusioni

Dalla Tabella emergono alcuni aspetti interessanti:

- **SGD puro (15 epoche).** Il modello ottiene una *Train Accuracy* pari a 82.16% e una *Valid Accuracy* di 76.79%. Si osserva una buona stabilità e un miglioramento costante sia in termini di loss sia di accuratezza.

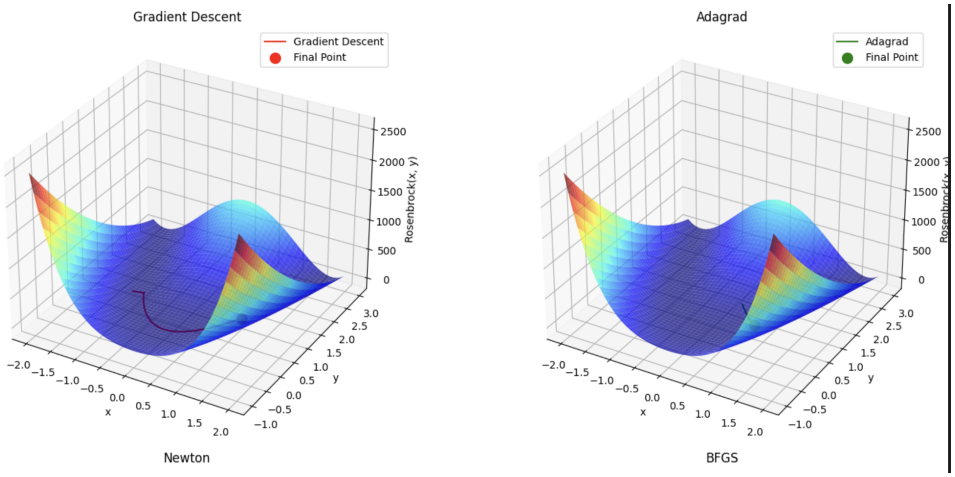


Figura 5.1: Esempio di ulteriore visualizzazione della funzione di Rosenbrock con metodi di ottimizzazione.

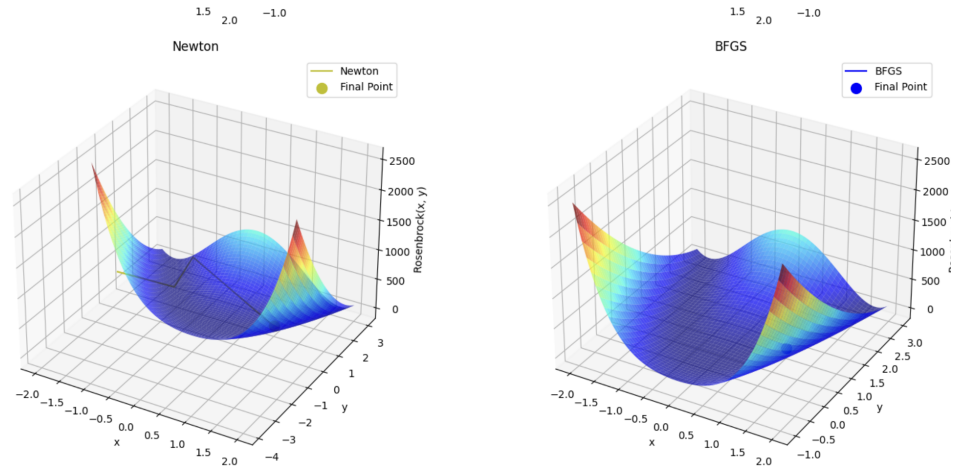


Figura 5.2: Esempio di ulteriore visualizzazione della funzione di Rosenbrock con metodi di ottimizzazione.

- **Warm Start (solo SGD, 5 epoche).** Questa prima fase di allenamento raggiunge il 72.90% di accuratezza sul training e il 67.57% sulla validazione. Benché inferiore rispetto a SGD dopo 15 epoche, la rete si avvicina già a un regime di convergenza di base.

- **Warm Start + Hessiano (Epoca 2).** Dopo sole 2 epoche di aggiornamenti Hessiani, il modello arriva al 91.97% di *Train Accuracy* e 78.52% di *Valid Accuracy*. Tale risultato evidenzia come l'uso di un metodo di seconda derivata (anche se approssimato) possa accelerare il calo della *Train Loss* e migliorare velocemente le performance in validazione, *almeno inizialmente*.

- **Warm Start + Hessiano (Epoca 10).** Proseguendo con ulteriori epoche di aggiornamento Hessiano, la *Train Accuracy* continua a salire (fino al 96.31%), ma la *Valid Accuracy* scende a 75.07% con una *Valid Loss* pari a 1.0174. Questo comportamento suggerisce un rischio di *overfitting* o di instabilità, dovuto all'aggiornamento

aggressivo tipico dei metodi di seconda derivata quando il modello si trova già in una regione di buon fit sul training set.

| Metodo | Epoca | Train Loss | Train Acc | Valid Loss | Valid Acc |
|---|---|---|---|---|---|
| Pure SGD | 15 | 0.4841 | 82.16% | 0.6326 | 76.79% |
| Warm Start (solo SGD) | 5 | 0.7091 | 72.90% | 0.8303 | 67.57% |
| Warm Start + Hessiano | 2* | 0.4402 | 91.97% | 0.6027 | 78.52% |
| Warm Start + Hessiano | 10* | 0.5084 | 96.31% | 1.0174 | 75.07% |

Tabella 5.1: Confronto dei risultati di training. Le epoche contrassegnate con * si riferiscono alle fasi del training con aggiornamenti Hessiani dopo il warm start.