Alma Mater Studiorum
Università di Bologna

Corso di Laurea in Ingegneria Biomedica

Fondamenti matematici dell'apprendimento nelle reti neurali

Tesi di Laurea in Analisi Matematica

Relatore:
Prof. Giovanni E. Comi

Correlatore:
Prof. Daniele Vigo

Candidato:
Thomas Gasser

Sessione Novembre 2025

Anno Accademico 2024 – 2025

Ringraziamenti

Vorrei ringraziare il Professor Comi per la grande pazienza e per il tempo che mi ha dedicato durante il lavoro di stesura della tesi. La cura con cui ha corretto ogni mio errore — non solo matematico, ma anche di punteggiatura e di grammatica (non erano pochi!) — mi ha davvero aiutato a migliorare, sia nella comprensione della matematica sia nell'uso della lingua italiana.

Soprattutto, Lo ringrazio per avermi trasmesso una forte passione per la matematica, prima durante le lezioni e poi mentre lavoravamo insieme sulla tesi.

Un sincero ringraziamento al Professor Daniele Vigo per la disponibilità e per i preziosi suggerimenti forniti nel corso della stesura della tesi.
La sua competenza ed esperienza hanno offerto spunti utili per approfondire e affinare diversi aspetti del lavoro, in particolare quelli relativi alla parte algoritmica.

Abstract

Questa tesi descrive i fondamenti matematici dell'apprendimento automatico, con un focus sulle reti neurali artificiali. Attraverso una trattazione rigorosa di algebra lineare, calcolo e ottimizzazione, dimostriamo come queste branche della matematica permettano di sviluppare meccanismi chiave, come la discesa del gradiente e la backpropagation. Il nucleo teorico poggia su risultati come il Teorema dell'Approssimazione Universale, che legittima l'architettura delle reti, e su condizioni di ottimalità e convergenza che ne garantiscono l'addestrabilità. L'obiettivo è quindi costruire un ponte tra l'astrazione matematica e l'efficacia pratica, fornendo una spiegazione teorica del funzionamento degli algoritmi di machine learning.

Indice

Introduzione

[1]

Motivazioni

Negli ultimi decenni l'Intelligenza Artificiale (AI) ed il Machine Learning (apprendimento automatico) hanno raggiunto traguardi straordinari, permettendo alle macchine di eseguire compiti un tempo riservati unicamente all'intelletto umano.
Dalla visione artificiale al riconoscimento del linguaggio naturale, i sistemi di apprendimento automatico sono diventati elementi chiave in innumerevoli applicazioni pratiche.

Al cuore di questo successo ci sono, tra l'altro, modelli matematici noti come reti neurali artificiali, che simulano il comportamento dei neuroni biologici per apprendere dai dati. Tuttavia, il comportamento "intelligente" di queste reti non nasce per magia: è il frutto di solida matematica e di algoritmi di ottimizzazione.
È la matematica che fornisce il linguaggio e gli strumenti per formulare, analizzare e risolvere i problemi di apprendimento. Questa tesi nasce dalla volontà di esplorare in profondità come la matematica, in particolare alcuni suoi settori chiave, permetta alle macchine di imparare, fornendo un ponte tra teoria astratta e innovazione tecnologica.

In ambito applicativo, spesso i modelli di machine learning e le reti neurali sono trattati come "scatole nere" i cui risultati empirici sono noti, ma i cui principi di funzionamento restano opachi.
Qui intendiamo "aprire" questa scatola nera, mettendo in luce i principi matematici che governano il funzionamento delle reti neurali. Comprendere tali principi non è solo un esercizio teorico, una comprensione approfondita consente di progettare architetture di rete migliori, di selezionare algoritmi di training appropriati e di garantire maggiore affidabilità e interpretabilità dei modelli.

Le motivazioni di questo lavoro risiedono dunque nell'obiettivo di fornire una trattazione unitaria e rigorosa dei fondamenti matematici dell'apprendimento automatico basato su reti neurali.

Panoramica dei fondamenti teorici

La tesi è articolata in più capitoli, organizzati in modo da accompagnare il lettore dai concetti matematici di base fino all'applicazione di tali concetti nell'ambito delle reti neurali artificiali e dei loro algoritmi di apprendimento.
Di seguito descriviamo brevemente il contenuto di ciascun capitolo.

[1] I codici utilizzati per la generazione delle immagini sono stati inizialmente realizzati con l'assistenza di ChatGPT, e successivamente modificati manualmente per ottenere il risultato desiderato.

Alla base dell'apprendimento automatico vi è un insieme di nozioni matematiche fondamentali, senza le quali sarebbe impensabile comprendere o migliorare algoritmi come la discesa del gradiente o la backpropagation.
In questa tesi ci concentriamo su tre pilastri teorici:

1. l'algebra lineare;
2. il calcolo differenziale;
3. l'ottimizzazione convessa.

Forniamo di seguito una breve panoramica del ruolo di ciascuno di essi nel contesto dell'apprendimento automatico.

Algebra lineare - Il linguaggio delle Reti Neurali L'algebra lineare fornisce il linguaggio delle reti neurali, fondamentale per rappresentare dati e modelli.
In un sistema di apprendimento, dati di input, caratteristiche, pesi sinaptici di una rete neurale e output sono tipicamente rappresentati tramite vettori e matrici.
Concetti di base come vettori e spazi vettoriali, matrici e trasformazioni lineari, autovalori e autovettori sono fondamentali per comprendere la struttura interna di un modello matematico.
Ad esempio, un livello di una rete neurale che combina linearmente gli ingressi può essere descritto come una trasformazione lineare

$$\mathbf{z}^{(l+1)} = \mathbf{W}^{(l+1)}\mathbf{a}^{(l)} + \mathbf{b}^{(l+1)},$$

dove $\mathbf{W}^{(l+1)}$ è una matrice dei pesi e $\mathbf{b}^{(l+1)}$ un vettore di bias, del rispettivo strato $l+1$.

Gli strumenti dell'algebra lineare (come la decomposizione spettrale delle matrici simmetriche, le norme e le forme quadratiche) sono cruciali anche per analizzare le proprietà delle funzioni obiettivo da ottimizzare: basti pensare che la curvatura locale di una funzione $f : \mathbb{R}^n \to \mathbb{R}$ è descritta dall'Hessiana (matrice delle derivate seconde), i cui autovalori forniscono indicazioni sulla natura di minimi o massimi locali.

Calcolo differenziale - La Base dell'Ottimizzazione Il calcolo differenziale (in una e più variabili) è il secondo ingrediente essenziale e fornisce gli strumenti per analizzare il comportamento locale delle funzioni obiettivo. Il concetto fondamentale è il gradiente $\nabla f(\mathbf{x})$, che indica la direzione di massima crescita della funzione f nel punto $\mathbf{x}$, il suo opposto $-\nabla f(\mathbf{x})$ indica la direzione di discesa più ripida, ovvero la direzione in cui la funzione decresce più rapidamente al primo ordine.

Il calcolo differenziale fornisce inoltre il teorema di Fermat (spesso noto con il suo nome inglese "First Order Necessary Condition"):

Teorema (Condizione del Primo Ordine - Teorema di Fermat)**.** *Sia $f : \mathbb{R}^n \to \mathbb{R}$ una funzione differenziabile. Se $\mathbf{x}^*$ è un punto di minimo locale di f, allora:*

$$\nabla f(\mathbf{x}^*) = \mathbf{0}.$$

In altre parole, i punti di massimo o minimo locale sono necessariamente punti critici (punti a gradiente nullo). Questo risultato costituisce la base teorica per qualsiasi algoritmo di ottimizzazione differenziabile.

La ricerca di minimi locali si traduce nella ricerca di soluzioni dell'equazione $\nabla f(\mathbf{x}) = 0$. Il calcolo differenziale in più variabili introduce anche strumenti come la direzionalità delle

derivate e la già citata matrice Hessiana, ovvero la matrice delle derivate parziali seconde, che permettono di sviluppare criteri di ottimalità del secondo ordine (ad esempio: se $\nabla f(\mathbf{x}^*) = 0$ e l'Hessiana $Hf(\mathbf{x}^*)$ è definito positivo, allora $\mathbf{x}^*$ è un minimo locale).
Questi concetti saranno centrali per comprendere la convergenza e la stabilità degli algoritmi di apprendimento.

Ottimizzazione Convessa - Il Quadro Teorico Ideale Il terzo pilastro è l'ottimizzazione convessa, un campo che studia problemi di minimizzazione di funzioni convesse su insiemi convessi. Sebbene l'addestramento di una rete neurale dia origine in generale a un problema non convesso (la funzione di costo tipicamente non è convessa a causa della natura non lineare delle reti), la teoria dell'ottimizzazione convessa fornisce importanti idee e garantisce proprietà desiderabili.

Teorema (Minimi Globali in Problemi Convessi)**.** *Sia $f : C \to \mathbb{R}$ una funzione convessa definita su un insieme convesso $C \subseteq \mathbb{R}^n$. Allora:*

1. *ogni minimo locale è anche minimo globale,*
2. *l'insieme dei minimi è convesso,*
3. *se f è strettamente convessa, il minimo globale è unico.*

Studiare tali casi più semplici è propedeutico a comprendere il comportamento di algoritmi di discesa del gradiente in scenari più complessi.

Reti Neurali Artificiali

Infine, unendo i tre ambiti sopraccitati, possiamo modellare formalmente una rete neurale artificiale.
La struttura fondamentale prevede:

- un layer di input che riceve il vettore dei dati $\mathbf{x}$ in ingresso;
- uno o più layer nascosti (i cosiddetti hidden layer), ciascuno dei quali calcola una trasformazione affine dei suoi input, parametrizzata da una matrice dei pesi $\mathbf{W}^{(l)}$ e un vettore bias $\mathbf{b}^{(l)}$, seguita da una funzione non lineare $f^{(l)}(\cdot)$;
- un layer di output che produce la predizione finale $\mathbf{y}$.

Una rete neurale feed-forward può essere vista come una composizione di funzioni

$$F(\mathbf{x}; \theta) = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}(\mathbf{x}),$$

dove ogni layer l calcola:

$$\begin{aligned} \mathbf{z}^{(l)} &= \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \\ \mathbf{a}^{(l)} &= f^{(l)}(\mathbf{z}^{(l)}), \end{aligned}$$

con $f^{(l)}$ funzione di attivazione non lineare.

L'output della rete è quindi una funzione complessa

$$F(\mathbf{x}; \theta),$$

dove θ rappresenta l'insieme di tutti i parametri $\{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}$ (pesi e bias) della rete.

Il Ruolo Cruciale della Non Linearità A questo punto dobbiamo enfatizzare esplicitamente il ruolo cruciale della non linearità introdotta dalle funzioni di attivazione: senza di essa, infatti, una rete con più layer si ridurrebbe a una semplice trasformazione affine equivalente ad un modello lineare.

Vediamolo in più dettaglio. Consideriamo una rete composta esclusivamente da trasformazioni affini. In quel caso, formalmente, ogni layer calcola

$$f^{(l)}(\mathbf{a}) = \mathbf{W}^{(l)}\mathbf{a} + \mathbf{b}^{(l)}.$$

In questo caso, è possibile dimostrare un risultato fondamentale: una rete neurale feed-forward di profondità arbitraria, priva di funzioni di attivazione non lineari, implementa globalmente una singola trasformazione affine.

La dimostrazione, che procede per induzione sul numero di layer, mostra come la composizione di L trasformazioni affini sia equivalente al prodotto delle rispettive matrici e a una combinazione lineare dei vettori di bias, collassando infine in un'unica operazione.

Vediamo il case baso della dimostrazione, cioè il caso con $L = 2$. Abbiamo la rete

$$F(\mathbf{x}) = f^{(2)}\Big(f^{(1)}(\mathbf{x})\Big).$$

Sostituendo le definizioni

$$\begin{aligned} F(\mathbf{x}) &= \mathbf{W}^{(2)}\Big(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\Big) + \mathbf{b}^{(2)} \\ &= \Big(\mathbf{W}^{(2)}\mathbf{W}^{(1)}\Big)\mathbf{x} + \Big(\mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)}\Big) \end{aligned}$$

Ponendo

$$\begin{aligned} \mathbf{W}' &= \mathbf{W}^{(2)}\mathbf{W}^{(1)}; \\ \mathbf{b}' &= \mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)}. \end{aligned}$$

otteniamo la trasformazione

$$\mathbf{y} = \mathbf{W}'\mathbf{x} + \mathbf{b}'.$$

Questo risultato ha un'implicazione immediata e profonda: senza non linearità, l'intera architettura di rete, per quanto profonda, equivale a un semplice modello di regressione lineare. Il "deep" in "deep learning" perderebbe ogni significato. È dunque la presenza delle funzioni di attivazione non lineari a permettere alla rete di comporre trasformazioni successive in modo non banale, aprendo la possibilità di rappresentare funzioni altamente non lineari e complesse.

Sorge spontanea una domanda cruciale: esiste una garanzia teorica che reti neurali di questo tipo siano effettivamente in grado di approssimare una vasta classe di funzioni? La risposta è fornita dal teorema fondamentale di questo capitolo: il Teorema dell'Approssimazione Universale.

Teorema (**Teorema dell'Approssimazione Universale**)**.** *Un Multi Layer Perceptron (MLP) con almeno uno layer nascosto e una funzione di attivazione non lineare è in grado di approssimare qualsiasi funzione continua su un insieme compatto di* $\mathbb{R}^n$*, con un numero sufficiente di neuroni nello layer nascosto.*

Questo risultato fornisce una giustificazione teorica della straordinaria adattabilità delle reti neurali: anche architetture apparentemente semplici possiedono, almeno in linea di principio, la capacità di rappresentare funzioni di complessità arbitraria. È importante osservare, tuttavia, che il teorema ha natura puramente esistenziale: garantisce che una certa configurazione di pesi esiste, ma non indica come trovarla, né quanti neuroni siano necessari, né con quale

difficoltà l'algoritmo di addestramento la possa raggiungere. Saranno proprio gli strumenti dell'ottimizzazione – in primis il metodo della discesa del gradiente e i suoi sviluppi – a fornire il collegamento pratico tra questa possibilità teorica e l'effettiva capacità di addestrare una rete per risolvere un problema reale.

La bontà della predizione è misurata da una funzione di costo $C(\theta)$, che quantifica l'errore commesso rispetto ai valori attesi $\hat{\mathbf{y}}$ su un insieme di dati di addestramento. Forme tipiche di $C(\theta)$ sono l'errore quadratico medio per problemi di regressione o l'entropia incrociata per problemi di classificazione.

L'apprendimento consiste nel trovare θ che minimizzi una funzione di costo $C(\theta)$. Qui entra in gioco l'ottimizzazione: minimizzare $C(\theta)$ significa risolvere un problema (potenzialmente non convesso) in uno spazio di dimensione elevata, problema affrontabile tramite metodi iterativi basati sul gradiente.

Discesa del gradiente: Algoritmo Fondamentale

L'addestramento delle reti neurali rappresenta una delle sfide computazionali più significative nell'ambito del machine learning. Questa complessità deriva dalla necessità di ottimizzare funzioni obiettivo altamente non lineari e non convesse, spesso in spazi con milioni di parametri. In questo contesto, il metodo della discesa del gradiente emerge come strumento fondamentale, fornendo un approccio iterativo ed efficiente per la minimizzazione di funzioni differenziabili.

Mentre in teoria potremmo risolvere il problema di ottimizzazione ponendo il gradiente uguale a zero e cercando una soluzione analitica, nella pratica questa approccio risulta impraticabile per diverse ragioni: le funzioni obiettivo delle reti neurali sono tipicamente non convesse, il numero di parametri è elevatissimo, e una soluzione analitica sarebbe computazionalmente intrattabile. È proprio in questo scenario che la discesa del gradiente dimostra tutta la sua utilità.

Il punto di partenza per comprendere la discesa del gradiente risiede nel significato geometrico del vettore gradiente. Data una funzione $f : \mathbb{R}^n \to \mathbb{R}$ differenziabile, il gradiente $\nabla f(\mathbf{x})$ in un punto $\mathbf{x}$ rappresenta la direzione di massima crescita locale della funzione.

Teorema (Direzione di Massima Discesa)**.** *Sia $f : \mathbb{R}^n \to \mathbb{R}$ una funzione differenziabile in $\mathbf{x}_0 \in \mathbb{R}^n$ con $\nabla f(\mathbf{x}_0) \neq \mathbf{0}$. Allora, tra tutte le direzioni unitarie $\mathbf{d} \in \mathbb{R}^n$ con $\|\mathbf{d}\| = 1$, la direzione che minimizza la derivata direzionale è*

$$\mathbf{d}^* = -\frac{\nabla f(\mathbf{x}_0)}{\|\nabla f(\mathbf{x}_0)\|}.$$

In questa direzione, la derivata direzionale di f in x_0 assume il valore $-\|\nabla f(\mathbf{x}_0)\|$.

La dimostrazione si basa sulla disuguaglianza di Cauchy-Schwarz. Per ogni direzione unitaria $\mathbf{d}$, abbiamo:

$$\nabla f(\mathbf{x}_0) \cdot \mathbf{d} \leq \|\nabla f(\mathbf{x}_0)\|\|\mathbf{d}\| = \|\nabla f(\mathbf{x}_0)\|,$$

dove l'uguaglianza vale se e solo se $\mathbf{d}$ è parallelo a $\nabla f(\mathbf{x}_0)$. La direzione di massima discesa si ottiene quindi considerando il vettore opposto al gradiente.

Questo risultato fondamentale ci dice che, localmente, la direzione più efficiente per far diminuire il valore della funzione è proprio quella opposta al gradiente. Questa intuizione geometrica costituisce il cuore dell'algoritmo di discesa del gradiente.

Per comprendere come sfruttare questa proprietà, consideriamo lo sviluppo di Taylor al primo ordine della funzione attorno al punto corrente $\mathbf{x}_0$:

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) + o(\|\mathbf{x} - \mathbf{x}_0\|).$$

Ora, se ci spostiamo nella direzione opposta al gradiente di una quantità $\alpha > 0$, otteniamo il punto:

$$\mathbf{x} = \mathbf{x}_0 - \alpha \nabla f(\mathbf{x}_0).$$

Sostituendo questo punto nell'espansione di Taylor e assumendo α sufficientemente piccolo da poter trascurare i termini di ordine superiore, otteniamo:

$$f(\mathbf{x}_0 - \alpha \nabla f(\mathbf{x}_0)) = f(\mathbf{x}_0) - \alpha \|\nabla f(\mathbf{x}_0)\|^2 + o(\alpha).$$

Poiché $\alpha > 0$ e $\|\nabla f(\mathbf{x}_0)\|^2 > 0$ (assumendo di non essere già in un punto stazionario), per α sufficientemente piccolo abbiamo con certezza:

$$f(\mathbf{x}_0 - \alpha \nabla f(\mathbf{x}_0)) < f(\mathbf{x}_0).$$

Questo dimostra rigorosamente che muovendoci nella direzione opposta al gradiente otteniamo effettivamente una riduzione del valore della funzione obiettivo.

Definizione (**Algoritmo di Discesa del Gradiente**)**.** *Dato un punto iniziale $\mathbf{x}_0 \in \mathbb{R}^n$ e una funzione obiettivo $f : \mathbb{R}^n \to \mathbb{R}$ differenziabile, l'algoritmo di discesa del gradiente procede iterativamente secondo la regola:*

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k), \quad k = 0, 1, 2, \ldots$$

dove $\alpha_k > 0$ è il learning rate (o passo di apprendimento) alla k-esima iterazione.

L'algoritmo può essere sintetizzato nei seguenti passi fondamentali:

1. **Inizializzazione**: Si scelgono un punto iniziale $\mathbf{x}_0$ e un learning rate $\alpha_0 > 0$.
2. **Iterazione**: Per $k = 0, 1, 2, \ldots$:
   - si calcola il gradiente: $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$,
   - si aggiorna la posizione: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k$,
   - si verifica il criterio di arresto.
3. **Terminazione**: Il processo si interrompe quando il criterio di arresto è soddisfatto.

La scelta del learning rate α_k è uno degli aspetti più critici per il successo dell'algoritmo. Questo parametro controlla l'ampiezza del passo che compiamo ad ogni iterazione lungo la direzione del gradiente.

La strategia più semplice consiste nell'utilizzare un learning rate costante $\alpha_k = \alpha$ per tutte le iterazioni. Questa approccio ha il vantaggio della semplicità implementativa, ma richiede una scelta accurata del parametro α. Un valore troppo piccolo porta a convergenza lenta, mentre un valore troppo grande può causare instabilità.

Nell'implementazione pratica dell'algoritmo, dobbiamo definire quando fermare il processo iterativo. La condizione teorica ideale $\nabla f(\mathbf{x}_k) = \mathbf{0}$ raramente è raggiungibile numericamente, quindi, fissata una soglia di accettabilità $\varepsilon > 0$, si utilizzano criteri approssimativi come:

1. Norma del gradiente: $\|\nabla f(\mathbf{x}_k)\| < \varepsilon$;
2. Variazione della funzione: $|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| < \varepsilon$;
3. Variazione dei parametri: $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon$.

Tutti i criteri riflettono la stessa idea fondamentale: man mano che l'algoritmo si avvicina a un punto stazionario, le variazioni nella posizione e nel valore della funzione obiettivo diventano sempre più piccole, indicando che il processo di ottimizzazione sta convergendo.

Backpropagation: Calcolo Efficiente del Gradiente

Dopo aver definito l'architettura delle reti neurali e la funzione di costo da minimizzare, il capitolo finale è dedicato agli algoritmi di training, cioè ai metodi per trovare i parametri θ ottimali che minimizzano la funzione di costo. Si riprende l'algoritmo di discesa del gradiente presentato in precedenza e lo si applica al contesto della rete neurale: qui θ denota l'insieme di tutte le matrici dei pesi e vettori dei bias della rete.

L'algoritmo di backpropagation gioca un ruolo importante nell'apprendimento delle reti.
L'algoritmo di backpropagation calcola efficientemente $\nabla_\theta C(\theta)$ applicando ricorsivamente la regola della catena. Le equazioni di aggiornamento sono:

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \alpha\, \boldsymbol{\delta}^{(l)} (\mathbf{a}^{(l-1)})^\top,$$
$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \alpha\, \boldsymbol{\delta}^{(l)},$$

dove $\boldsymbol{\delta}^{(l)}$ è l'errore retropropagato dal layer l.

Questa struttura matematica fornisce un ponte solido tra teoria astratta e implementazione pratica, mostrando come strumenti matematici classici permettano di risolvere problemi moderni di apprendimento automatico.

In sintesi, la struttura della tesi accompagna il lettore dall'astrazione matematica alla concretezza dell'implementazione algoritmica, mostrando passo per passo "come la matematica permette alle macchine di imparare."

Capitolo 1

Algebra Lineare

1.1 Nozioni di Base

Chiamiamo una collezione di oggetti un "insieme", e i vari oggetti che lo compongono vengono detti "elementi".

Definizione 1.1.1 (**Campo**)**.** Sia $K \subseteq \mathbb{C}$ un insieme, allora K viene detto un campo se soddisfa le seguenti condizioni:

1. se $x, y \in K$, allora $x + y \in K$ e $xy \in K$;
2. se $x \in K$, allora $-x \in K$; se inoltre $x \neq 0$, allora anche $x^{-1} \in K$;
3. $0 \in K$ e $1 \in K$.

Se K è un campo, allora gli elementi di K vengono detti scalari.

Esempi tipici di un campo sono l'insieme dei numeri reali $\mathbb{R}$ e l'insieme dei numeri complessi $\mathbb{C}$.

Definizione 1.1.2 (**Spazio vettoriale**)**.** Dati un insieme $\mathbf{V}$ e un campo di scalari K, $\mathbf{V}$ è uno spazio vettoriale su K, i cui elementi si dicono vettori, se sono definite le seguenti operazioni:

1. *Somma di vettori*: per ogni $\mathbf{u}, \mathbf{v} \in \mathbf{V}$, allora $\mathbf{u} + \mathbf{v} \in \mathbf{V}$;
2. *Moltiplicazione per scalare*: per ogni $a \in K$ e $\mathbf{v} \in \mathbf{V}$, allora $a\mathbf{v} \in \mathbf{V}$.

Inoltre valgono le seguenti proprietà per ogni $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbf{V}$ e $a, b \in K$:

- *Associatività*:
$$(\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w});$$
- *Vettore nullo*:
esiste un elemento $\mathbf{0} \in \mathbf{V}$ tale che:
$$\mathbf{u} + \mathbf{0} = \mathbf{0} + \mathbf{u} = \mathbf{u};$$
- *Inverso*:
per ogni $\mathbf{u} \in \mathbf{V}$ esiste $-\mathbf{u} \in \mathbf{V}$ con:
$$\mathbf{u} + (-\mathbf{u}) = \mathbf{0};$$

- *Commutatività*:

$$\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u};$$

- *Distributività della moltiplicazione per scalari rispetto alla somma di vettori*:

$$a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v};$$

- *Distributività della moltiplicazione per scalari rispetto alla somma di scalari*:

$$(a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v};$$

- *Associatività della moltiplicazione scalare*:

$$(ab)\mathbf{v} = a(b\mathbf{v});$$

- *Identità*:

$$1 \cdot \mathbf{v} = \mathbf{v}.$$

Definizione 1.1.3 (**Spazio** $\mathbb{R}^n$)**.** Sia $n \in \mathbb{N}$. Lo spazio $\mathbb{R}^n$ è definito come il prodotto cartesiano di n copie dell'insieme dei numeri reali:

$$\mathbb{R}^n := \underbrace{\mathbb{R} \times \mathbb{R} \times \cdots \times \mathbb{R}}_{n \text{ volte}}.$$

Un elemento di $\mathbb{R}^n$ è una n-upla ordinata di numeri reali,

$$\mathbf{x} = (x_1, x_2, \ldots, x_n), \quad x_i \in \mathbb{R} \text{ per } i = 1, \ldots, n.$$

Ai fini dei calcoli, rappresentiamo $\mathbf{x}$ come vettore colonna:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

Il corrispondente vettore riga, detto trasposto di $\mathbf{x}$, è denotato da

$$\mathbf{x}^\top = \begin{pmatrix} x_1, \ldots, x_n \end{pmatrix} \in (\mathbb{R}^n)^\top.$$

Osservazione 1.1.4. L'insieme $\mathbb{R}^n$ è uno spazio vettoriale su $\mathbb{R}$. Per vederlo definiamo le operazioni della definizione 1.1.2 componente per componente. Supponiamo che $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ e $a \in \mathbb{R}$. Allora la somma di vettori è data da

$$\mathbf{x} + \mathbf{y} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} := \begin{pmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{pmatrix} \in \mathbb{R}^n,$$

e la moltiplicazione per uno scalare è data da

$$a\mathbf{x} = a \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} := \begin{pmatrix} ax_1 \\ \vdots \\ ax_n \end{pmatrix} \in \mathbb{R}^n.$$

Poiché per la definizione 1.1.1, $\mathbb{R}$ è un campo, le proprietà di somma e prodotto in $\mathbb{R}$ garantiscono automaticamente che queste operazioni soddisfino tutti gli assiomi di spazio vettoriale. Questo spazio vettoriale viene definito come "spazio euclideo reale n-dimensionale $\mathbb{R}^n$". Ovviamente questo vale anche per qualsiasi altro campo. Quindi $\mathbb{C}^n$ è uno spazio vettoriale su $\mathbb{C}$ e $\mathbb{Q}^n$ su $\mathbb{Q}$, per dare qualche esempio.

Definizione 1.1.5 (**Base di uno spazio vettoriale**)**.** Sia $\mathbf{V}$ uno spazio vettoriale su un campo K. Un insieme di vettori $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ in $\mathbf{V}$ si dice una base di $\mathbf{V}$ se soddisfa le seguenti due condizioni:

1. è un insieme di vettori linearmente indipendenti;
2. genera lo spazio vettoriale $\mathbf{V}$.

Equivalentemente, ogni vettore $\mathbf{v} \in \mathbf{V}$ può essere espresso in modo unico come combinazione lineare

$$\mathbf{v} = a_1 v_1 + a_2 v_2 + \cdots + a_n v_n, \quad \text{con } a_1, a_2, \ldots, a_n \in K.$$

Gli scalari $a_1, a_2, \ldots, a_n$ sono dette le coordinate di v rispetto alla base $\{v_1, v_2, \ldots, v_n\}$.

Definizione 1.1.6 (**Dimensione**)**.** Sia $\mathbf{V}$ uno spazio vettoriale che ammette una base finita. Il numero di elementi in una base di $\mathbf{V}$ è detto dimensione di V e viene indicato con $\dim \mathbf{V}$.

Definizione 1.1.7 (**Base canonica di** $\mathbb{R}^n$)**.** Lo spazio vettoriale $\mathbb{R}^n$ ha una base particolarmente importante, detta base canonica (o base standard). Essa è costituita dai vettori:

$$\mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \ldots, \mathbf{e}_n = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix},$$

dove il vettore $\mathbf{e}_i$ ha 1 nella i-esima componente e 0 in tutte le altre.

Osservazione 1.1.8. La base canonica di $\mathbb{R}^n$ è costituita da n vettori. Di conseguenza, per definizione, la dimensione di $\mathbb{R}^n$ è n.

1.2 Prodotto scalare

Un'operazione fondamentale in $\mathbb{R}^n$ è il prodotto scalare fra due vettori. Questo prodotto è un modo per quantificare il grado di ortogonalità o parallelismo.

Definizione 1.2.1 (**Prodotto scalare**)**.** Dati $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, il prodotto scalare fra $\mathbf{x}$ e $\mathbf{y}$ è la quantità scalare:

$$\mathbf{x} \cdot \mathbf{y} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n = \sum_{j=1}^{n} x_j y_j.$$

Notazione 1.2.2. Il prodotto scalare tra due vettori può essere rappresentato in modi diversi, ciascuno con i propri vantaggi in contesti specifici, per il momento usiamo

$$\mathbf{x} \cdot \mathbf{y}.$$

In seguito, una volta introdotto il prodotto fra matrici useremo anche

$$\mathbf{x}^\top \mathbf{y}.$$

Definizione 1.2.3 (**Vettori ortogonali**)**.** Diciamo che due vettori non nulli $\mathbf{x}$ e $\mathbf{y}$ sono ortogonali se il loro prodotto scalare è zero, ovvero

$$\mathbf{x} \cdot \mathbf{y} = 0.$$

Osservazione 1.2.4. Se abbiamo due vettori ortogonali, nel piano generato da questi vettori c'è un angolo di 90° tra loro.

Proposizione 1.2.5 (**Proprietà del prodotto scalare**)**.** *Per ogni* $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$ *e* $a, b \in \mathbb{R}$ *valgono le seguenti proprietà:*

1. *Commutatività:*
$$\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{n} x_i \cdot y_i;$$
2. *Proprietà distributiva:*
$$(a\mathbf{x} + b\mathbf{y}) \cdot \mathbf{z} = a(\mathbf{x} \cdot \mathbf{z}) + b(\mathbf{y} \cdot \mathbf{z});$$
3. *Non negatività e non degenerazione:*
$$\mathbf{x} \cdot \mathbf{x} \geq 0 \quad e \quad \mathbf{x} \cdot \mathbf{x} = 0 \iff \mathbf{x} = 0.$$

Teorema 1.2.6 (**Disuguaglianza di Cauchy-Schwarz**)**.** *Per ogni* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ *abbiamo*

$$|\mathbf{x} \cdot \mathbf{y}| \leq \|\mathbf{x}\| \, \|\mathbf{y}\|,$$

o, equivalentemente

$$-\|\mathbf{x}\| \, \|\mathbf{y}\| \leq \mathbf{x} \cdot \mathbf{y} \leq \|\mathbf{x}\| \, \|\mathbf{y}\|,$$

dove l'uguaglianza $\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\| \, \|\mathbf{y}\|$ *vale se e solo se* $\mathbf{x} = 0$ *o* $\mathbf{y} = \lambda\mathbf{x}$ *per qualche* $\lambda \geq 0$*, mentre l'uguaglianza* $\mathbf{x} \cdot \mathbf{y} = -\|\mathbf{x}\| \, \|\mathbf{y}\|$ *vale se e solo se* $\mathbf{x} = 0$ *o* $\mathbf{y} = \lambda\mathbf{x}$ *per qualche* $\lambda \leq 0$*.*

1.3 Norme L^p

La funzione che usiamo per misurare la grandezza di un vettore viene chiamata norma. Le norme sono funzioni che associano a un vettore un numero reale non negativo. A livello intuitivo, la norma di un vettore $\mathbf{x}$ che parte dall'origine e arriva al punto $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ rappresenta un tipo di distanza dall'origine al punto $\mathbf{x}$.

Definizione 1.3.1 (**Norma**)**.** Una norma è una qualunque funzione $\|\cdot\| : \mathbb{R}^n \to [0, +\infty)$ che soddisfi le seguenti tre proprietà per ogni $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ e ogni scalare $\alpha \in \mathbb{R}$:

1. **Positività:**
$$\|\mathbf{x}\| \geq 0 \quad \text{e} \quad \|\mathbf{x}\| = 0 \iff \mathbf{x} = 0,$$
ovvero, la norma è sempre positiva oppure nulla, per il vettore nullo;
2. **Positiva omogeneità:**
$$\|\alpha\mathbf{x}\| = |\alpha| \, \|\mathbf{x}\|,$$
se un vettore viene moltiplicato per uno scalare, la norma si scala proporzionalmente con il valore assoluto dello scalare;

3. **Disuguaglianza triangolare:**

$$\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|,$$

la norma della somma non supera la somma delle norme.

Questa definizione è molto astratta e generale, in pratica esistono delle norme che sono più intuitive nelle applicazioni, dato che misurano la grandezza di un vettore in modi che sembrano più "naturali" dal punto di vista geometrico o computazionale. Le norme più comuni sono quelle legate alle somme di potenze delle componenti del vettore, chiamate norme L^p.

Definizione 1.3.2 (**Norma L^p**)**.** La norma L^p in $\mathbb{R}^n$ è definita per $p \in [1, +\infty)$ come

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}} \quad \text{per ogni} \quad \mathbf{x} \in \mathbb{R}^n.$$

In questa tesi ci serviranno principalmente i casi $p = 1$ e $p = 2$ delle norme L^p. Di conseguenza, le ridefiniamo a parte.

Definizione 1.3.3 (**Norma di Manhattan**)**.** La norma di Manhattan, o norma L^1, è il caso di $p = 1$ delle norme L^p, ed è la somma dei valori assoluti delle componenti del vettore, ovvero per ogni $\mathbf{x} \in \mathbb{R}^n$ poniamo

$$\|\mathbf{x}\|_1 = \sum_{j=1}^{n} |x_j| = |x_1| + |x_2| + \cdots + |x_n|.$$

Definizione 1.3.4 (**Norma Euclidea**)**.** La norma euclidea, o norma L^2, è il caso di $p = 2$ delle norme L^p, ed è definita come la radice quadrata della somma dei quadrati delle componenti del vettore, ovvero, per ogni $\mathbf{x} \in \mathbb{R}^n$, poniamo

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^{n} x_j^2} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

Osservazione 1.3.5. La norma euclidea è la generalizzazione diretta del teorema di Pitagora a spazi con più di due dimensioni ed è il modo più naturale in cui definiamo la distanza tra due punti nello spazio euclideo.

Per semplicità matematica e computazionale si usa spesso la norma L^2 al quadrato:

$$\|\mathbf{x}\|_2^2 = \sum_{j=1}^{n} x_j^2 = x_1^2 + x_2^2 + \cdots + x_n^2.$$

Osservazione 1.3.6. Applicando la definizione di prodotto scalare (Definizione 1.2.1) al caso $\mathbf{y} = \mathbf{x}$, si ha

$$\mathbf{x} \cdot \mathbf{x} = \sum_{i=1}^{n} x_i^2.$$

Dunque,

$$\|\mathbf{x}\|^2 = \mathbf{x} \cdot \mathbf{x}.$$

Osservazione 1.3.7. Se prendiamo la norma L^2 al quadrato, non abbiamo più una norma, dato che la proprietà dell'omogeneità non è più soddisfatta:

$$\|\alpha\mathbf{x}\|_2^2 = \alpha^2 \cdot \|\mathbf{x}\|_2^2 \neq \alpha \|\mathbf{x}\|_2^2 \qquad \forall\ \alpha \notin \{-1, 0, 1\}.$$

Si osserva che, oltre alle norme L^p definite per $1 \leq p < +\infty$, è possibile estendere la definizione anche al caso limite $p = +\infty$. In tale caso si ottiene la cosiddetta norma del massimo, che rappresenta il valore limite della norma L^p quando p tende all'infinito.

Definizione 1.3.8 (**Norma del massimo**)**.** La norma L^∞ è definita come il massimo dei valori assoluti delle componenti del vettore, ovvero, per ogni $\mathbf{x} \in \mathbb{R}^n$,

$$\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|.$$

Definizione 1.3.9 (**Equivalenza tra norme**)**.** Siano $\|\cdot\|_a$ e $\|\cdot\|_b$ due norme definite sullo stesso spazio vettoriale $\mathbf{V}$. Diciamo che queste norme sono equivalenti se esistono due costanti reali positive $C_1, C_2 > 0$ tali che

$$C_1\|\mathbf{x}\|_b \leq \|\mathbf{x}\|_a \leq C_2\|\mathbf{x}\|_b, \quad \forall \mathbf{x} \in \mathbf{V}.$$

In altre parole, l'equivalenza tra norme non implica che i valori delle due norme coincidano esattamente, ma che il rapporto fra di essi sia uniformemente controllato sia dall'alto che dal basso. Questo significa che, nella pratica, possiamo stimare o confrontare grandezze o errori usando due qualsiasi norme equivalenti senza perdere generalità, visto che la stima rimane valida a meno di un fattore moltiplicativo fisso.

Osservazione 1.3.10. In particolare, tutte le norme L^p su $\mathbb{R}^n$ sono equivalenti fra loro. Più in generale, tutte le norme su uno spazio vettoriale a dimensione finita, come $\mathbb{R}^n$, sono equivalenti.

Un concetto fondamentale che combina le nozioni di ortogonalità e norma è quello di ortonormalità.

Definizione 1.3.11 (**Vettori ortonormali**)**.** Sia $\mathbf{V}$ uno spazio vettoriale reale. Due vettori $\mathbf{u}, \mathbf{v} \in V$ si dicono ortonormali se sono:

1. ortogonali tra loro, ovvero $\mathbf{u} \cdot \mathbf{v} = 0$;
2. di norma unitaria, ovvero $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$.

Osservazione 1.3.12. Due vettori ortogonali e non nulli formano tra loro un angolo di 90°. L'ortonormalità aggiunge il requisito che ciascun vettore abbia lunghezza unitaria, il ché semplifica molti calcoli.

1.4 Algebra matriciale

Consideriamo ora un nuovo tipo di oggetto, le matrici.

Definizione 1.4.1 (**Matrice**)**.** Siano $m, n \in \mathbb{N}$, con $m, n \geq 1$. Definiamo la matrice $\mathbf{A}$ di dimensione $m \times n$ (con m righe e n colonne) come la tabella ordinata degli elementi a_{ij}, con $i = 1, \ldots, m$ e $j = 1, \ldots, n$, ovvero

$$\mathbf{A} = [a_{ij}]_{\substack{i=1,\ldots,m \\ j=1,\ldots,n}} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix} \in \mathbb{R}^{m \times n}.$$

Indichiamo con $\mathbb{R}^{m \times n}$ l'insieme di tutte le matrici reali $m \times n$.

Osservazione 1.4.2. Possiamo anche vedere una matrice $\mathbf{A} \in \mathbb{R}^{m\times n}$ come un elemento dello spazio vettoriale $\mathbb{R}^{mn}$ dato che può essere interpretata come un vettore con mn componenti. Di conseguenza $\mathbb{R}^{m\times n}$ è uno spazio vettoriale, con operazioni di somma e prodotto per scalare definiti componente per componente in maniera analoga a quanto visto in precedenza per i vettori.

Definiamo adesso nel dettaglio queste operazioni.

Definizione 1.4.3 (**Somma di matrici**)**.** Siano $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m\times n}$. Definiamo la loro somma come la matrice $\mathbf{C} = \mathbf{A} + \mathbf{B}$, anch'essa di dimensione $m \times n$, in cui gli elementi c_{ij} sono dati da:

$$c_{ij} = a_{ij} + b_{ij}.$$

In altre parole, la somma delle matrici si fa elemento per elemento, ovvero:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \dots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \dots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \dots & a_{mn} + b_{mn} \end{bmatrix}. \tag{1.4.1}$$

Osservazione 1.4.4. La somma di matrici è commutativa e associativa, ovvero per ogni $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{m\times n}$:

$$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A},$$

e

$$\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}.$$

Definizione 1.4.5 (**Moltiplicazione di una matrice per uno scalare**)**.** Siano $\mathbf{A} \in \mathbb{R}^{m\times n}$ e $\lambda \in \mathbb{R}$. Definiamo il loro prodotto $\lambda\mathbf{A}$ moltiplicando ogni elemento di $\mathbf{A}$ per λ, ovvero:

$$\lambda\mathbf{A} = [\lambda a_{ij}]_{m\times n}.$$

Osservazione 1.4.6. La moltiplicazione di una matrice per uno scalare è associativa rispetto alla somma, sia di matrici

$$\lambda(\mathbf{A} + \mathbf{B}) = \lambda\mathbf{A} + \lambda\mathbf{B},$$

che di scalari

$$(\lambda + \mu)\mathbf{A} = \lambda\mathbf{A} + \mu\mathbf{A},$$

che rispetto alla moltiplicazione fra scalari

$$\lambda(\mu\mathbf{A}) = (\lambda\mu)\mathbf{A}.$$

per ogni $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m\times n}$ e $\lambda, \mu \in \mathbb{R}$.

Le regole per la somma tra matrici e la moltiplicazione per scalari sono relativamente semplici e intuitive. Tuttavia, quando si tratta della moltiplicazione tra matrici, le regole diventano più complesse.

Moltiplicazione tra matrici

La moltiplicazione tra due matrici è una delle operazioni più importanti dell'algebra lineare. Non vengono semplicemente moltiplicati i singoli elementi delle matrici (come invece accade nel caso del "prodotto di Hadamard", che definiremo più avanti), ma viene utilizzata una procedura più complessa.

Il calcolo di ogni elemento della matrice prodotto coinvolge una colonna intera di una matrice e una riga intera dell'altra. L'operazione è quindi definita solo per matrici di dimensioni specifiche, ovvero la prima matrice deve avere un numero di colonne pari al numero di righe della seconda matrice.

Definizione 1.4.7 (**Moltiplicazione tra matrici**)**.** Siano $\mathbf{A} \in \mathbb{R}^{m\times n}$ e $\mathbf{B} \in \mathbb{R}^{n\times p}$. Il loro prodotto $\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m\times p}$ è definito come la matrice avente per elementi i termini

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

In altre parole, il prodotto fra queste matrici è dato dalla matrice $m \times p$ le cui componenti sono i prodotti scalari fra i vettori della riga i-esima di $\mathbf{A}$ con la colonna j-esima di $\mathbf{B}$, che hanno infatti la stessa dimensione n.

Osservazione 1.4.8. Nel caso particolare $m = p = 1$ abbiamo il tra vettori, che può infatti essere interpretato come una moltiplicazione tra un vettore colonna e un vettore riga, che ci dà quindi una matrice 1×1, ovvero uno scalare.

Infatti, dati

$$\mathbf{A} = \mathbf{a}^\top = \begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix},$$

e

$$\mathbf{B} = \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix},$$

abbiamo

$$\mathbf{AB} = \mathbf{a}^\top \cdot \mathbf{b} = \begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} = \sum_{j=1}^{n} a_j b_j.$$

Si osservi che, affinché il prodotto tra due matrici sia definito in entrambi gli ordini, è necessario che esse siano quadrate e della stessa dimensione. Tuttavia, anche in tal caso il prodotto non risulta, in generale, commutativo. Più precisamente, per ogni $n \geq 2$ esistono matrici $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ tali che $\mathbf{AB} \neq \mathbf{BA}$.

Osservazione 1.4.9. Anche se le matrici sono quadrate, il prodotto non è commutativo, ovvero, per ogni $n \geq 2$, esistono $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ tali che

$$\mathbf{AB} \neq \mathbf{BA}.$$

Vediamo un esempio.

Esempio 1.4.10. Siano $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{2\times 2}$ date da

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \text{e} \quad \mathbf{B} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}.$$

Semplici calcoli mostrano che

$$\mathbf{AB} = \begin{bmatrix} 1\times 5 + 2\times 7 & 1\times 6 + 2\times 8 \\ 3\times 5 + 4\times 7 & 3\times 6 + 4\times 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

e

$$\mathbf{BA} = \begin{bmatrix} 5\times 1 + 6\times 3 & 5\times 2 + 6\times 4 \\ 7\times 1 + 8\times 3 & 7\times 2 + 8\times 4 \end{bmatrix} = \begin{bmatrix} 23 & 34 \\ 31 & 46 \end{bmatrix}.$$

Osservazione 1.4.11. In forma matriciale, l'identità

$$\|\mathbf{x}\|^2 = \mathbf{x} \cdot \mathbf{x},$$

può essere riscritta come

$$\|\mathbf{x}\|^2 = \mathbf{x}^\top \mathbf{x}.$$

Osservazione 1.4.12 (**Prodotto di una matrice e un vettore**)**.** Un altro caso particolare rilevante è il prodotto matrice $\mathbf{A} \in \mathbb{R}^{m\times n}$ per un vettore colonna $\mathbf{x} \in \mathbb{R}^n$, che risulta in una matrice di dimensione $(m \times n) \times (n \times 1) = (m \times 1)$, quindi un vettore colonna della dimensione dei vettori riga della matrice. Infatti abbiamo

$$\mathbf{Ax} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \end{bmatrix} \in \mathbb{R}^{m\times 1}.$$

Si può definire anche il prodotto elemento per elemento di due matrici; questa operazione è diversa dalla definizione di moltiplicazione fra matrici presentata in precedenza. Questo prodotto viene chiamato prodotto di Hadamard.

Definizione 1.4.13 (**Prodotto di Hadamard**)**.** Date $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m\times n}$, definiamo il prodotto $\mathbf{A} \odot \mathbf{B}$ come

$$\mathbf{A} \odot \mathbf{B} = [a_{ij} b_{ij}]_{m\times n}.$$

Questo significa che gli elementi della nuova matrice sono semplicemente dati dal prodotto di ciascun elemento corrispondente delle due matrici.

Osservazione 1.4.14. Poiché la moltiplicazione scalare è commutativa, anche il prodotto di Hadamard lo è.

Matrici importanti

Definizione 1.4.15 (**Matrice nulla**)**.** L'elemento nullo di $\mathbb{R}^{m\times n}$ è la matrice nulla, ovvero la matrice che ha tutti gli elementi uguali a zero:

$$\mathbf{0} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}.$$

Definizione 1.4.16 (**Matrice Diagonale**)**.** Una matrice diagonale è una matrice quadrata $\mathbf{D} \in \mathbb{R}^{n\times n}$ con $d_{ij} = 0$ per ogni $i \neq j$. In altre parole, solo gli elementi sulla diagonale principale (che parte dall'elemento d_{11} e va all'elemento d_{nn}) della matrice possono essere diversi da zero.

Osservazione 1.4.17. Il prodotto con una matrice diagonale risulta particolarmente semplice da calcolare. Sia

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \in \mathbb{R}^{m\times n}$$

una matrice qualsiasi, e

$$\mathbf{D} = \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_m \end{bmatrix} \in \mathbb{R}^{m\times m}$$

una matrice diagonale. Allora si ha:

$$\mathbf{DA} = \begin{bmatrix} d_1a_{11} & d_1a_{12} & \dots & d_1a_{1n} \\ d_2a_{21} & d_2a_{22} & \dots & d_2a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_ma_{m1} & d_ma_{m2} & \dots & d_ma_{mn} \end{bmatrix},$$

cioè la moltiplicazione a sinistra per $\mathbf{D}$ corrisponde a moltiplicare la i-esima riga di $\mathbf{A}$ per d_i.

Analogamente, se $\mathbf{A} \in \mathbb{R}^{m\times n}$ e

$$\mathbf{D} = \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_n \end{bmatrix} \in \mathbb{R}^{n\times n},$$

allora

$$\mathbf{AD} = \begin{bmatrix} d_1a_{11} & d_2a_{12} & \dots & d_na_{1n} \\ d_1a_{21} & d_2a_{22} & \dots & d_na_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_1a_{m1} & d_2a_{m2} & \dots & d_na_{mn} \end{bmatrix},$$

cioè la moltiplicazione a destra per $\mathbf{D}$ corrisponde a moltiplicare la j-esima colonna di $\mathbf{A}$ per d_j.

Definizione 1.4.18 (**Matrice Identità**)**.** La matrice identità è una matrice diagonale in cui gli elementi della diagonale principale sono tutti 1; ovvero:

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}.$$

Osservazione 1.4.19. La matrice identità è l'elemento neutro del prodotto, infatti, se $\mathbf{I} \in R^{n\times n}$, $\mathbf{A} \in \mathbb{R}^{m\times n}$ e $\mathbf{B} \in \mathbb{R}^{n\times p}$, allora

$$\mathbf{AI} = \mathbf{A} \qquad \text{e} \qquad \mathbf{IB} = \mathbf{B}.$$

Osservazione 1.4.20. Come vedremo in seguito, la matrice identità ci permette di definire l'inversa di una matrice $\mathbf{A}^{-1}$.

Definizione 1.4.21 (**Matrice trasposta**)**.** La trasposta di una matrice $\mathbf{A} \in \mathbb{R}^{m\times n}$ è la matrice $\mathbf{A}^\top \in \mathbb{R}^{n\times m}$, data elemento per elemento da

$$(\mathbf{A}^\top)_{i,j} = a_{j,i}.$$

In altre parole, si scambiano righe e colonne.

Osservazione 1.4.22. In particolare, nel caso di una matrice quadrata, la trasposta è data dall'immagine speculare della matrice lungo la diagonale principale.

Esempio 1.4.23. Consideriamo la matrice $\mathbf{A} \in \mathbb{R}^{3\times 2}$:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}.$$

La sua trasposta è la matrice $\mathbf{A}^\top \in \mathbb{R}^{2\times 3}$ ottenuta scambiando righe e colonne:

$$\mathbf{A}^\top = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}.$$

Osservazione 1.4.24. Il vettore colonna e il vettore riga definiti in precedenza sono uno il trasposto dell'altro. In questa tesi $\mathbf{v}$ indica il vettore colonna, quindi $\mathbf{v}^\top$ è il vettore riga.

Osservazione 1.4.25. Uno scalare invece può essere pensato come matrice con una sola riga e una sola colonna, quindi l'operazione della trasposta non lo cambia.

Proposizione 1.4.26. *Siano* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m\times n}$ *e* $\lambda \in \mathbb{R}$*, allora valgono le seguenti proprietà della trasposizione:*

1. $\boldsymbol{A}^{TT} = \boldsymbol{A}$*;*
2. $(\mathbf{A} + \mathbf{B})^\top = \mathbf{A}^\top + \mathbf{B}^\top$*;*
3. $(\lambda\mathbf{A})^\top = \lambda\mathbf{A}^\top$*;*
4. $(\mathbf{AB})^\top = \mathbf{B}^\top\mathbf{A}^\top$.

Definizione 1.4.27 (**Matrice Simmetrica**)**.** Una matrice simmetrica è una matrice quadrata $\mathbf{A} \in \mathbb{R}^{n\times n}$ che coincide con la sua trasposta, ovvero

$$\mathbf{A}^\top = \mathbf{A}.$$

Definizione 1.4.28 (**Matrice Ortogonale**)**.** Una matrice ortogonale è una matrice quadrata $\mathbf{A} \in \mathbb{R}^{n\times n}$ che soddisfa

$$\mathbf{A}^\top \mathbf{A} = \mathbf{A}\mathbf{A}^\top = \mathbf{I}.$$

Lemma 1.4.29. *Sia* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *una matrice ortogonale. Allora, per ogni vettore* $\mathbf{x} \in \mathbb{R}^n$*, la trasformazione* $\mathbf{y} = \mathbf{A}^\top \mathbf{x}$ *preserva la norma euclidea*

$$\|\mathbf{y}\| = \|\mathbf{A}^\top \mathbf{x}\| = \|\mathbf{x}\|$$

Dimostrazione. Calcoliamo la norma al quadrato di $\mathbf{y}$

$$\|\mathbf{y}\|^2 = \mathbf{y}^\top \mathbf{y} = (\mathbf{A}^\top \mathbf{x})^\top (\mathbf{A}^\top \mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{A}^\top \mathbf{x}.$$

Poiché $\mathbf{A}$ è ortogonale, vale $\mathbf{A}\mathbf{A}^\top = \mathbf{I}$, quindi

$$\|\mathbf{y}\|^2 = \mathbf{x}^\top \mathbf{I}\mathbf{x} = \|\mathbf{x}\|^2.$$

Prendiamo la radice quadrata e otteniamo

$$\|\mathbf{y}\| = \|\mathbf{x}\|.$$

□

Inversa di una matrice

Consideriamo il seguente sistema lineare:

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

dove

- $\mathbf{A} \in \mathbb{R}^{n\times n}$ è la matrice quadrata dei coefficienti,
- $\mathbf{b} \in \mathbb{R}^n$ è il vettore dei termini noti,
- $\mathbf{x} \in \mathbb{R}^n$ è il vettore delle incognite;

Ora proviamo a risolvere questa equazione. Se questa fosse un'equazione algebrica della forma $Ax = b$ con $A, x, b \in \mathbb{R}$, quindi quantità scalari, e supponessimo che $A \neq 0$, allora si potrebbe risolvere per x semplicemente dividendo per A:

$$x = \frac{b}{A} = \frac{1}{A} b = A^{-1} b.$$

Però non si può dividere per una matrice, quindi non è possibile risolvere un sistema lineare in questo modo. Esiste un concetto che è l'equivalente della divisione: la moltiplicazione per l'inversa.

L'inversa di una matrice è quella matrice che, se moltiplicata a destra o a sinistra per A, produce la matrice identità.

Definizione 1.4.30 (**Matrice inversa**)**.** Sia $\mathbf{A} \in \mathbb{R}^{n\times n}$ una matrice quadrata. Definiamo la matrice inversa di $\mathbf{A}$, $\mathbf{A}^{-1} \in \mathbb{R}^{n\times n}$, come la matrice che soddisfa

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I} = \mathbf{A}\mathbf{A}^{-1}.$$

Una matrice quadrata $\mathbf{A} \in \mathbb{R}^{n\times n}$ è detta **invertibile** (o non singolare) se e solo se ammette una matrice inversa $\mathbf{A}^{-1}$.

Osservazione 1.4.31. La matrice inversa, se esiste, è unica.

Se esiste la matrice inversa, possiamo risolvere il sistema lineare nel seguente modo:

$$\mathbf{Ax} = \mathbf{b} \iff \mathbf{A}^{-1}\mathbf{Ax} = \mathbf{A}^{-1}\mathbf{b} \iff \mathbf{Ix} = \mathbf{A}^{-1}\mathbf{b} \iff \mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

In questi passaggi abbiamo approfittato del fatto che $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ e che qualsiasi vettore o matrice moltiplicato per la matrice identità rimane invariato.

Osservazione 1.4.32. Una matrice $\mathbf{A} \in \mathbb{R}^{n\times n}$ è ortogonale se e solo se

$$\mathbf{A}^{-1} = \mathbf{A}^{\top}. \tag{1.4.2}$$

Proposizione 1.4.33 (Caratterizzazione delle matrici ortogonali). *Una matrice $\mathbf{A} \in \mathbb{R}^{n\times n}$ è ortogonale se e solo se le sue righe (o, equivalentemente, le sue colonne) formano un insieme di vettori ortonormali, ovvero vettori mutuamente ortogonali e di norma euclidea pari a 1.*

Dimostrazione. Sia $\mathbf{A} = [\mathbf{c}_1|\mathbf{c}_2|\dots|\mathbf{c}_n] \in \mathbb{R}^{n\times n}$, con $\mathbf{c}_i \in \mathbb{R}^n$ le colonne della matrice, e assumiamo che siano ortonormali. Considerando il prodotto di $\mathbf{A}$ con la sua trasposta $\mathbf{A}^{\top}$, otteniamo una matrice i cui elementi sono i prodotti scalari tra le colonne

$$(\mathbf{A}^{\top}\mathbf{A})_{ij} = \mathbf{c}_i \cdot \mathbf{c}_j.$$

Quindi:

- se $i = j$, si ha $\mathbf{c}_i \cdot \mathbf{c}_i = \|\mathbf{c}_i\|^2 = 1$;
- se $i \neq j$, si ha $\mathbf{c}_i \cdot \mathbf{c}_j = 0$.

La matrice risultante $\mathbf{A}^{\top}\mathbf{A}$ ha quindi 1 sulla diagonale principale e 0 altrove, cioè

$$\mathbf{A}^{\top}\mathbf{A} = \mathbf{I} \iff \mathbf{A}^{-1} = \mathbf{A}^{\top}.$$

Viceversa, se $\mathbf{A}$ è ortogonale, per definizione vale $\mathbf{A}^{\top}\mathbf{A} = \mathbf{I}$. In tal caso, l'elemento (i, j) di tale prodotto è dato da

$$(\mathbf{A}^{\top}\mathbf{A})_{ij} = \mathbf{c}_i \cdot \mathbf{c}_j.$$

Poiché $\mathbf{A}^{\top}\mathbf{A} = \mathbf{I}$, segue che:

- $(\mathbf{A}^{\top}\mathbf{A})_{ii} = 1 \implies \|\mathbf{c}_i\| = 1$ per ogni i;
- $(\mathbf{A}^{\top}\mathbf{A})_{ij} = 0$ per $i \neq j \implies \mathbf{c}_i \perp \mathbf{c}_j$.

Pertanto, le colonne di $\mathbf{A}$ risultano ortonormali. □

1.5 Trasformazione matriciale

In algebra lineare, una matrice non è solo una tabella di numeri, ma può essere interpretata come la rappresentazione di una trasformazione dello spazio. Una matrice $\mathbf{A} \in \mathbb{R}^{m\times n}$ definisce una regola che accetta un vettore in input (in $\mathbb{R}^n$) e produce un nuovo vettore in output (in $\mathbb{R}^m$), agendo come operatore che applica rotazioni, dilatazioni, contrazioni e proiezioni.

Definizione 1.5.1 (Trasformazione). Una trasformazione $T : \mathbb{R}^n \to \mathbb{R}^m$ è una funzione che associa a ogni vettore $\mathbf{x} \in \mathbb{R}^n$ un vettore $T(\mathbf{x}) \in \mathbb{R}^m$.

Definizione 1.5.2 (**Trasformazione lineare**)**.** Una trasformazione $T : \mathbb{R}^n \to \mathbb{R}^m$ è detta lineare se preserva le operazioni di somma vettoriale e moltiplicazione per scalare. Formalmente, per ogni $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ e per ogni $\lambda \in \mathbb{R}$ devono valere:

1. Additività:
$$T(\mathbf{x} + \mathbf{y}) = T(\mathbf{x}) + T(\mathbf{y});$$
2. Omogeneità:
$$T(\lambda \mathbf{x}) = \lambda T(\mathbf{x}).$$

Queste due proprietà implicano che la trasformazione mantiene l'origine fissa e che le linee rette rimangono linee rette.

Definizione 1.5.3 (**Trasformazione matriciale**)**.** Sia $\mathbf{A} \in \mathbb{R}^{m \times n}$ una matrice. La trasformazione matriciale associata ad $\mathbf{A}$ è la funzione $T_{\mathbf{A}} : \mathbb{R}^n \to \mathbb{R}^m$ definita da

$$T_A(\mathbf{x}) = \mathbf{A}\mathbf{x}.$$

Ogni trasformazione matriciale è lineare. Viceversa, come mostra il teorema successivo, ogni trasformazione lineare può essere rappresentata come una trasformazione matriciale.

Teorema 1.5.4 (**Rappresentazione matriciale di una trasformazione lineare**)**.** *Sia* $T : \mathbb{R}^n \to \mathbb{R}^m$ *una trasformazione lineare. Allora esiste una matrice* $\mathbf{A}_T \in \mathbb{R}^{m \times n}$ *tale che*

$$T(\mathbf{x}) = \mathbf{A}_T \mathbf{x} \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

La matrice $\mathbf{A}_T$ *è chiamata matrice rappresentativa di* T *(rispetto alla base canonica) e si costruisce ponendo come sue colonne le immagini dei vettori della base canonica di* $\mathbb{R}^n$*:*

$$\mathbf{A}_T = \left[T(\mathbf{e}_1) | T(\mathbf{e}_2) | \dots | T(\mathbf{e}_n) \right].$$

Dimostrazione. Sia $\mathbf{x} \in \mathbb{R}^n$ un vettore. Poiché $\{\mathbf{e}_i\}_{i=1}^n$ è la base canonica, possiamo scrivere $\mathbf{x}$ come combinazione lineare dei vettori $\{\mathbf{e}_i\}_{i=1}^n$:

$$\mathbf{x} = x_1 \mathbf{e}_1 + x_2 \mathbf{e}_2 + \cdots + x_n \mathbf{e}_n,$$

dove $x_1, \dots, x_n$ sono le coordinate di $\mathbf{x}$. Applicando T e usando la linearità otteniamo

$$T(\mathbf{x}) = x_1 T(\mathbf{e}_1) + x_2 T(\mathbf{e}_2) + \cdots + x_n T(\mathbf{e}_n),$$

che è proprio la definizione di moltiplicazione matrice per vettore

$$T(\mathbf{x}) = \mathbf{A}\mathbf{x} \quad \forall\, \mathbf{x} \in \mathbb{R}^n.$$

□

1.6 Autodecomposizione

L'autodecomposizione, o decomposizione spettrale, è una tecnica che permette di scrivere una matrice quadrata $\mathbf{A}$ in termini dei suoi autovalori e autovettori. Questa rappresentazione semplifica operazioni complesse come il calcolo di potenze di matrici e la risoluzione di equazioni lineari.

Definizione 1.6.1 (**Autovalori e autovettori**)**.** Sia $\mathbf{A} \in \mathbb{R}^{n\times n}$ una matrice quadrata. Un vettore non nullo $\mathbf{v} \in \mathbb{R}^n$, $\mathbf{v} \neq 0$, è detto autovettore di $\mathbf{A}$ se esiste uno scalare $\lambda \in \mathbb{R}$ tale che

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}. \tag{1.6.1}$$

Lo scalare λ è detto autovalore di $\mathbf{A}$ associato all'autovettore $\mathbf{v}$.

Osservazione 1.6.2. È possibile interpretare una matrice $\mathbf{A}$ come un operatore che agisce sui vettori, trasformandoli in altri vettori. In generale, tale trasformazione può comprendere:

- rotazioni, che modificano la direzione del vettore,
- ridimensionamenti, che ne alterano la norma,
- oppure combinazioni più complesse di entrambe.

Gli autovettori individuano invece direzioni "speciali": lungo queste direzioni l'azione di $\mathbf{A}$ è particolarmente semplice, perché non cambia la direzione del vettore ma soltanto la sua norma (eventualmente invertendone il verso). L'equazione fondamentale

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

esprime proprio questo fatto: l'effetto di $\mathbf{A}$ su $\mathbf{v}$ consiste nel moltiplicarlo per uno scalare. Tale scalare λ, detto autovalore, rappresenta il fattore di scala (positivo o negativo) lungo la direzione dell'autovettore.

Proposizione 1.6.3. *Sia $\mathbf{A} \in \mathbb{R}^{n\times n}$ una matrice quadrata. Se $\mathbf{v}_1, \ldots, \mathbf{v}_k$ sono autovettori di $\mathbf{A}$ associati ad autovalori distinti $\lambda_1, \ldots, \lambda_k$, allora i vettori $\mathbf{v}_1, \ldots, \mathbf{v}_k$ sono linearmente indipendenti.*

Questa proprietà è di fondamentale importanza, poiché garantisce che, nel caso in cui una matrice $\mathbf{A}$ ammetta n autovalori distinti, i corrispondenti autovettori formino una base di $\mathbb{R}^n$.

In tale base, l'azione di $\mathbf{A}$ risulta particolarmente semplice: la matrice associata diventa diagonale. Questo porta al concetto di diagonalizzazione, che consente di semplificare notevolmente i calcoli sfruttando autovalori e autovettori.

Definizione 1.6.4 (**Diagonalizzazione**)**.** Una matrice quadrata $\mathbf{A} \in \mathbb{R}^{n\times n}$ è detta diagonalizzabile se esiste una base di $\mathbb{R}^n$ costituita da autovettori di $\mathbf{A}$. In tal caso, la matrice $\mathbf{A}$ può essere scritta nella forma:

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1},$$

dove:

- $\mathbf{V} \in \mathbb{R}^{n\times n}$ è la matrice degli autovettori, i cui vettori colonna sono gli autovettori di $\mathbf{A}$;
- $\mathbf{\Lambda} \in \mathbb{R}^{n\times n}$ è una matrice diagonale i cui elementi sono gli autovalori $\lambda_1, \lambda_2, \ldots, \lambda_n$ corrispondenti;
- $\mathbf{V}^{-1}$ è l'inversa della matrice $\mathbf{V}$, che esiste grazie all'indipendenza lineare degli autovettori.

Osservazione 1.6.5 (**Utilità computazionale**)**.** La decomposizione agli autovalori semplifica notevolmente il calcolo delle potenze di matrici. Per ogni intero positivo k:

$$\mathbf{A}^k = (\mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1})^k = \mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^{-1}.$$

Il vantaggio computazionale risiede nel fatto che $\mathbf{\Lambda}^k$ si calcola immediatamente elevando ciascun elemento diagonale alla k-esima potenza, evitando la moltiplicazione ripetuta di matrici.

Per dimostrare questo fatto, notiamo che $\mathbf{V}$ e $\mathbf{V}^{-1}$ si annullano a coppie. Per vederlo prendiamo l'esempio

$$\mathbf{A}^2 = (\mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1})(\mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1})$$

Vediamo subito che il prodotto nel mezzo $\mathbf{V}^{-1} \cdot \mathbf{V} = \mathbf{I}$ perché sono l'inversa dell'altra. Lo stesso ragionamento si ripete per qualsiasi potenza.

Nel caso di matrici già diagonali, il calcolo degli autovalori e autovettori è immediato: gli autovalori di una matrice diagonale sono semplicemente gli elementi che si trovano sulla sua diagonale principale, e gli autovettori corrispondenti sono i vettori della base canonica. Tuttavia, per matrici non diagonali, il procedimento diventa più complesso.

Fortunatamente, esiste un metodo sistematico per trovare gli autovalori basato sul calcolo della determinante.

Definizione 1.6.6 (**Determinante**)**.** Il determinante è la funzione $det : \mathbb{R}^{m\times n} \to \mathbb{R}$ definita per induzione su n nel seguente modo: data $\mathbf{A} \in \mathbb{R}^{n\times n}$

- se $n = 1$, allora $det(\mathbf{A}) := a_{11}$;
- se $n \geq 2$ allora

$$det(\mathbf{A}) := \sum_{i=1}^{n} a_{i1}(-1)^{i+1} det(\mathbf{A}_{i1}),$$

dove $\mathbf{A}_{i1}$ è la matrice quadrata $(n-1) \times (n-1)$ che si ottiene eliminando la prima colonna e la riga i-esima.

Proposizione 1.6.7. *Sia $\mathbf{A} \in \mathbb{R}^{n\times n}$. Allora, $\lambda \in \mathbb{R}$ è un autovalore di $\mathbf{A}$ se e solo se soddisfa*

$$\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = 0.$$

Osservazione 1.6.8. Di conseguenza chiamiamo l'espressione

$$p(\lambda) = det(\mathbf{A} - \lambda\mathbf{I})$$

il polinomio caratteristico. Gli zeri di questo polinomio sono gli autovalori di $\mathbf{A}$. Questo polinomio ha grado n, quindi, per il teorema fondamentale dell'algebra, abbiamo al massimo n autovalori.

Una volta determinati gli autovalori, per trovare i corrispondenti autovettori si procede sostituendo ciascun autovalore λ nell'equazione (1.6.1). Questa può essere riscritta nella forma

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{v} = 0,$$

dove si ricercano le soluzioni non banali, ossia i vettori $\mathbf{v} \neq 0$.

Nello studio delle applicazioni, in particolare nell'analisi delle funzioni tramite la matrice Hessiana, compaiono in modo naturale le matrici simmetriche. Le matrici simmetriche sono quindi di particolare interesse; i loro autovalori hanno un significato geometrico chiaro e permettono di caratterizzare proprietà fondamentali della funzione, come convessità o presenza di punti di massimo e minimo.

Teorema 1.6.9 (**Teorema Spettrale**)**.** *Sia $\mathbf{A} \in \mathbb{R}^{n\times n}$ una matrice simmetrica. Allora:*

1. $\mathbf{A}$ *è diagonalizzabile, ovvero esistono una matrice diagonale* Λ *e una matrice ortogonale* $\mathbf{V}$ *tali che*
$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\,\boldsymbol{V}^{\top}; \tag{1.6.2}$$

2. *tutti gli autovalori di* $\mathbf{A}$ *sono numeri reali.*

3. *esiste una base ortonormale di* $\mathbb{R}^n$ *costituita da autovettori di* $\mathbf{A}$*.*

Osservazione 1.6.10. L'equazione (1.6.2) è una diretta conseguenza della Proposizione 1.4.33 e dell'Osservazione 1.4.32.

Questo teorema ci fornisce gli strumenti per una comprensione geometrica della forma quadratica e quindi del comportamento della matrice Hessiana.

Definizione 1.6.11 (**Forma quadratica**)**.** Data una matrice simmetrica $\mathbf{A} \in \mathbb{R}^{n\times n}$, la forma quadratica associata a $\mathbf{A}$ è una funzione che, per ogni vettore $\mathbf{x} \in \mathbb{R}^n$, associa uno scalare definito come:
$$\mathbf{Q}_A(\mathbf{x}) = \mathbf{x}^{\top}\mathbf{A}\mathbf{x}.$$

Grazie al Teorema Spettrale, possiamo analizzare la forma quadratica. Sostituendo la decomposizione $\mathbf{A} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$ otteniamo
$$\mathbf{Q}_A(\mathbf{x}) = \mathbf{x}^{\top}(\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T)\mathbf{x}.$$
Definendo il vettore $\mathbf{y} = \mathbf{V}^{\top}\mathbf{x}$, che rappresenta le coordinate di $\mathbf{x}$ nel sistema di riferimento ruotato i cui assi sono gli autovettori di $\mathbf{A}$, la forma quadratica assume una forma semplicissima:
$$\mathbf{Q}_A(\mathbf{x}) = \mathbf{y}^{\top}\boldsymbol{\Lambda}\mathbf{y} = \lambda_1 y_1^2 + \lambda_2 y_2^2 + \cdots + \lambda_n y_n^2.$$
Questa equazione ci offre un'intuizione geometrica immediata. Ci dice che la forma quadratica $\mathbf{Q}_A(\mathbf{x})$ non è altro che una combinazione ponderata dei quadrati delle coordinate del nuovo sistema di riferimento, e che i pesi di questa combinazione sono gli autovalori $\lambda_1, \lambda_2, \ldots, \lambda_n$. Il segno e la ampiezza di questi autovalori determinano completamente la curvatura della funzione associata alla matrice $\mathbf{A}$ lungo le sue direzioni principali (gli autovettori).
Nello specifico, abbiamo i seguenti casi:

- **Segno positivo** ($\mathbf{Q}_A(\mathbf{x}) > 0$): indica una curvatura convessa ("a U") nella direzione di $\mathbf{x}$; la funzione tende a salire muovendosi in quella direzione;
- **Segno negativo** ($\mathbf{Q}_A(\mathbf{x}) < 0$): indica una curvatura concava nella direzione di $\mathbf{x}$; la funzione tende a scendere muovendosi in quella direzione;
- **Valore assoluto grande:** indica una curvatura molto accentuata (pendio ripido);
- **Valore assoluto piccolo:** indica una curvatura lieve (pendio dolce o regione quasi piatta).

Questa interpretazione ci porta a classificare il comportamento della forma quadratica, e quindi della matrice $\mathbf{A}$, in base ai segni dei suoi autovalori.
Perciò, è utile introdurre la nozione di segno di una matrice simmetrica.

Definizione 1.6.12 (**Segno di una matrice**)**.** Siano $\mathbf{A} \in \mathbb{R}^{n\times n}$ una matrice simmetrica e
$$\mathbf{Q}_A(\mathbf{x}) = \mathbf{x}^{\top}\mathbf{A}\mathbf{x}$$
la forma quadratica ad essa associata. Diciamo che la matrice A è

1. **definita positiva** se $\mathbf{Q}_A(\mathbf{x}) > 0$ per ogni $\mathbf{x} \neq \mathbf{0}$;
2. **definita negativa** se $\mathbf{Q}_A(\mathbf{x}) < 0$ per ogni $\mathbf{x} \neq \mathbf{0}$;
3. **semi-definita positiva** se $\mathbf{Q}_A(\mathbf{x}) \geq 0$ per ogni $\mathbf{x} \in \mathbb{R}^n$;
4. **semi-definita negativa** se $\mathbf{Q}_A(\mathbf{x}) \leq 0$ per ogni $\mathbf{x} \in \mathbb{R}^n$;
5. **non definita** se $\mathbf{Q}_A(\mathbf{x})$ assume valori sia positivi sia negativi al variare di $\mathbf{x}$.

Teorema 1.6.13 (**Criterio di Sylvester o teorema spettrale per il segno**). *Sia* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *una matrice simmetrica, e siano* $\lambda_1, \ldots, \lambda_n$ *i suoi autovalori. Allora valgono le seguenti equivalenze:*

1. $\mathbf{A}$ *è definita positiva* $\iff$ $\lambda_i > 0$ *per ogni* i*;*
2. $\mathbf{A}$ *è definita negativa* $\iff$ $\lambda_i < 0$ *per ogni* i*;*
3. $\mathbf{A}$ *è semi-definita positiva* $\iff$ $\lambda_i \geq 0$ *per ogni* i*;*
4. $\mathbf{A}$ *è semi-definita negativa* $\iff$ $\lambda_i \leq 0$ *per ogni* i*;*
5. $\mathbf{A}$ *è non definita* $\iff$ *gli autovalori hanno segni diversi, cioè esistono* $i, k \in \{1, \ldots, n\}$ *tali che* $\lambda_i > 0$ *e* $\lambda_k < 0$*.*

Notazione 1.6.14. Spesso, per rappresentare la definita di una matrice vengono usati i seguenti simboli:

- $\mathbf{A} \succ 0$ se è definita positiva;
- $\mathbf{A} \succeq 0$ se è semi-definita positiva;
- $\mathbf{A} \prec 0$ se è definita negativa;
- $\mathbf{A} \preceq 0$ se è semi-definita negativa.

Teorema 1.6.15 (**Invertibilità di matrici definite**). *Sia* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *una matrice simmetrica e definita positiva (o definita negativa). Allora* $\mathbf{A}$ *è invertibile.*

Dimostrazione. Si supponga per assurdo che $\mathbf{A}$ non sia invertibile. Allora esiste un vettore non nullo $\mathbf{x} \neq \mathbf{0}$ tale che

$$\mathbf{A}\mathbf{x} = \mathbf{0}.$$

Ma se $\mathbf{A}$ è definita positiva, per definizione deve valere che

$$\mathbf{y}^\top \mathbf{A}\mathbf{y} > 0 \qquad \forall\, \mathbf{y} \neq \mathbf{0}.$$

Tuttavia, scegliendo quindi $\mathbf{y} = \mathbf{x}$ e usando $\mathbf{A}\mathbf{x} = \mathbf{0}$, si ottiene

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} = \mathbf{x}^\top \mathbf{0} = \mathbf{0},$$

il ché contraddice l'ipotesi.
Pertanto, un tale vettore $\mathbf{x}$ non può esistere e $\mathbf{A}$ è invertibile. La dimostrazione per una matrice definita negativa è analoga. □

Teorema 1.6.16 (**Decomposizione spettrale dell'inversa**). *Sia* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *una matrice simmetrica e definita positiva, con decomposizione spettrale data da*

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top,$$

dove $\mathbf{V}$ *è una matrice ortogonale (*$\mathbf{V}^{-1} = \mathbf{V}^\top$*) i cui vettori colonna sono gli autovettori ortonormali di* $\mathbf{A}$*, e*

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} = diag(\lambda_1, \dots, \lambda_n)$$

è la matrice diagonale degli autovalori.

Allora l'inversa di $\mathbf{A}$ *è data da:*

$$\mathbf{A}^{-1} = \mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{V}^\top,$$

dove $\mathbf{\Lambda}^{-1} = diag(\lambda_1^{-1}, \lambda_2^{-1}, \dots, \lambda_n^{-1})$.

Dimostrazione. La dimostrazione segue direttamente dalla sostituzione e dalle proprietà delle matrici ortogonali:

$$\mathbf{A}^{-1} = (\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top)^{-1} = (\mathbf{V}^\top)^{-1}\mathbf{\Lambda}^{-1}\mathbf{V}^{-1} = \mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{V}^\top.$$

L'inversa $\mathbf{\Lambda}^{-1}$ esiste perché tutti gli autovalori λ_i sono non nulli, perché sono tutti positivi per il Teorema 1.6.13. □

Osservazione 1.6.17. Questo risultato ha notevoli implicazioni pratiche. Una volta calcolata la decomposizione spettrale di $\mathbf{A}$, il calcolo dell'inversa diventa estremamente efficiente, poiché richiede solo l'inversione degli elementi diagonali di $\mathbf{\Lambda}$. Inoltre, questo metodo è numericamente stabile per matrici simmetriche definite positive ben condizionate e costituisce la base di molte applicazioni in ottimizzazione e calcolo numerico.

Capitolo 2

Calcolo Differenziale

2.1 Nozioni di topologia e continuità delle funzioni

Definizione 2.1.1 (**Funzioni continue**)**.** Siano $E \subseteq \mathbb{R}^n$ un insieme, $f : E \to \mathbb{R}$ una funzione e $\mathbf{x}_0 \in E$. La funzione f è continua in $\mathbf{x}_0$ se per ogni $\varepsilon > 0$ esiste un $\delta > 0$ tale che, per ogni $\mathbf{x} \in E$ con $\|\mathbf{x} - \mathbf{x}_0\| < \delta$, abbiamo $|f(\mathbf{x}) - f(\mathbf{x_0})| < \varepsilon$.

Diciamo che $f : E \to \mathbb{R}$ è continua in E se è continua in ogni punto $\mathbf{x}_0 \in E$. Denotiamo con $C(E)$, o equivalentemente $C^0(E)$, lo spazio delle funzioni continue su E.

Definizione 2.1.2 (**Insieme aperto**)**.** Un insieme $A \subseteq \mathbb{R}^n$ è aperto in $\mathbb{R}^n$ se per ogni $\mathbf{x} \in A$ esiste $r > 0$ tale che la palla aperta con centro $\mathbf{x}$ e raggio r

$$B(\mathbf{x}, r) = \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y} - \mathbf{x}\| < r\}$$

è contenuta in A.
In altre parole, un insieme è aperto se ogni suo punto ha una piccola regione tutto intorno che sta ancora nell'insieme.

Definizione 2.1.3 (**Insieme compatto**)**.** Un insieme E si dice:

- chiuso se il suo complementare $\mathbb{R}^n \setminus E$ è aperto;
- limitato se esiste un raggio $R > 0$ tale che $E \subseteq B_R(0)$.

Un insieme $E \subseteq \mathbb{R}^n$ si dice compatto se è chiuso e limitato.

Nonostante il calcolo differenziale richieda di considerare come domini principalmente insiemi aperti, le funzioni continue godono di importanti proprietà su un'altra famiglia di domini, ovvero gli insiemi compatti.

2.2 Limiti di funzioni

I limiti per funzioni in più variabili sono un'estensione del concetto di limite per le funzioni di una variabile. Analogamente, lo studio dei limiti in più variabili si trova alla base dei concetti sia di continuità negli insiemi aperti che di derivabilità (e quindi di derivata parziale e gradiente). Si può definire il concetto di limite per funzioni: $f : E \subseteq \mathbb{R}^n \to \mathbb{R}^m$ utilizzando le norme, che permettono di definire la distanza in spazi multidimensionali.

Definizione 2.2.1 (**Limite**)**.** Siano $f : E \subseteq \mathbb{R}^n \to \mathbb{R}^m$ e $\mathbf{x}_0 \in \mathbb{R}^n$ un punto di accumulazione di E, ovvero tale che $(E \cap B(\mathbf{x}_0, r)) \setminus \{\mathbf{x}_0\} \neq \emptyset$ per ogni $r > 0$. Si dice che:

$$\lim_{\mathbf{x} \to \mathbf{x}_0} f(\mathbf{x}) = \mathbf{L} \in \mathbb{R}^m$$

se per ogni $\varepsilon > 0$ esiste un $\delta > 0$ tale che per ogni $\mathbf{x} \in E$ con

$$0 < ||\mathbf{x} - \mathbf{x}_0|| < \delta \qquad \text{abbiamo} \qquad ||f(\mathbf{x}) - \mathbf{L}|| < \varepsilon,$$

dove $||\mathbf{x} - \mathbf{x}_0||$ è la distanza tra $\mathbf{x}$ e $\mathbf{x}_0$ in $\mathbb{R}^n$, e $||f(\mathbf{x}) - \mathbf{L}||$ la distanza tra $f(\mathbf{x})$ e $\mathbf{L}$ in $\mathbb{R}^m$.

Osservazione 2.2.2. Nella definizione precedente, $\mathbf{x}_0$ deve essere un punto di accumulazione di E, cioè ogni palla centrata in $\mathbf{x}_0$ deve contenere punti di E diversi da $\mathbf{x}_0$. Se l'insieme E è aperto, tutti i suoi punti sono automaticamente punti di accumulazione, quindi la condizione è soddisfatta.

Osservazione 2.2.3. La scelta della norma (euclidea, Manhattan, del massimo, ecc.) è irrilevante per la definizione di limite, poiché in $\mathbb{R}^n$ tutte le norme sono equivalenti (si veda l'Osservazione 1.3.9).

Osservazione 2.2.4. A differenza del caso con una sola variabile, nei limiti in più variabili dobbiamo considerare tutti i possibili percorsi con cui ci si può avvicinare al punto $\mathbf{x}_0$: lungo rette, curve, parabole, ecc.

Affinché il limite esista, il valore di f deve tendere allo stesso risultato indipendentemente dal percorso scelto. Se il valore del limite dipende dal percorso scelto, allora il limite non esiste.

Osservazione 2.2.5. Siano $E \subseteq \mathbb{R}^n$ un insieme aperto, f una funzione vettoriale $f : E \to \mathbb{R}^m$ e $\mathbf{x}_0 \in E$. Allora f è continua in $\mathbf{x}_0$ se il limite di $f(\mathbf{x})$ per $\mathbf{x}$ che tende a $\mathbf{x}_0$ esiste ed è uguale a $f(\mathbf{x}_0)$, ovvero

$$\lim_{\mathbf{x} \to \mathbf{x}_0} f(\mathbf{x}) = f(\mathbf{x}_0).$$

2.3 Derivate

2.3.1 Richiamo: una variabile

Definizione 2.3.1 (**Derivabilità in un punto**)**.** Sia $I \subseteq \mathbb{R}$ un intervallo e $f : I \to \mathbb{R}$ una funzione. Allora diciamo che f è derivabile in un punto x_0 se esiste ed è finito il limite

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

e tale limite si chiama derivata prima di f in x_0.

Osservazione 2.3.2. La derivabilità in un punto x_0 significa che, localmente, la funzione è ben approssimata da una retta tangente

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + o(x - x_0).$$

La derivata rappresenta la pendenza di questa retta.

Proposizione 2.3.3. *Se $f : I \to \mathbb{R}$ è derivabile in x_0, allora f è continua in x_0. Il viceversa non è in generale vero.*

2.3.2 Funzioni di più variabili

Da qui fino alla fine di questo capitolo supponiamo che $\Omega \subseteq \mathbb{R}^n$ sia un insieme aperto.

Richiamiamo la Definizione 1.1.7 e la notazione $\mathbf{e}_j$ per il j-esimo vettore della base canonica di $\mathbb{R}^n$, cioè

$$\mathbf{e}_j = (0, \ldots, 1, \ldots, 0),$$

dove l'1 occupa la j-esima posizione.

Derivate parziali

Le derivate parziali rappresentano una naturale estensione del concetto di derivata alle funzioni di più variabili. Mentre la derivata di una funzione di una sola variabile misura la variazione di una funzione rispetto alla sua variabile indipendente, le derivate parziali misurano come la funzione varia rispetto a ciascuna variabile, mantenendo costanti le altre variabili.

Definizione 2.3.4 (**Derivata parziale**)**.** Siano $f : \Omega \to \mathbb{R}$, $\mathbf{x}_0 \in \Omega$ e $j \in \{1, ..., n\}$. La derivata parziale di f rispetto a x_j (j-esima componente del vettore delle variabili) in $\mathbf{x}_0$ è il limite

$$\frac{\partial f}{\partial x_j}(\mathbf{x}_0) = \lim_{h \to 0 \text{ in } \mathbb{R}} \frac{f(\mathbf{x}_0 + h\mathbf{e}_j)}{h} = \lim_{h \to 0} \frac{f(x_{0,1}, \ldots, x_{0,j} + h, \ldots, x_{0,n}) - f(\mathbf{x}_0)}{h},$$

se esiste finito.

Osservazione 2.3.5 (Interpretazione geometrica)**.** La derivata parziale $\frac{\partial f}{\partial x_j}(\mathbf{x}_0)$ rappresenta il tasso di cambiamento di f in $\mathbf{x}_0$ al variare della sola j-esima variabile, mantenendo costanti tutte le altre. Geometricamente, il grafico di f è un insieme in $\mathbb{R}^{n+1}$. Per visualizzare la derivata parziale, si considera la curva ottenuta intersecando il grafico con il piano

$$\{(x_1, x_2, \ldots, x_n, x_{n+1}) \in \mathbb{R}^{n+1} : x_i = x_{0,i} \qquad \text{per ogni} \qquad i \neq j\}.$$

La pendenza di questa curva nel punto $(\mathbf{x}_0, f(\mathbf{x}_0))$ coincide con $\frac{\partial f}{\partial x_j}(\mathbf{x}_0)$.

Osservazione 2.3.6. Per calcolare la derivata parziale rispetto a x_j, possiamo trattare tutte le altre $n - 1$ variabili come costanti e applicare le regole di derivazione per funzioni di una variabile.

Definizione 2.3.7 (**Gradiente**)**.** Siano $f : \Omega \to \mathbb{R}$ e $x_0 \in \Omega$. Se esistono tutte le derivate parziali in $\mathbf{x}_0$, ovvero

$$\exists \quad \frac{\partial f}{\partial x_j}(\mathbf{x}_0) \quad \forall\, j \in \{1; ...; n\},$$

allora il vettore riga che le ha come elementi è il gradiente di f in $\mathbf{x}_0$,

$$\nabla f(\mathbf{x}_0) = \left(\frac{\partial f}{\partial x_1}(\mathbf{x}_0), \frac{\partial f}{\partial x_2}(\mathbf{x}_0), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{x}_0) \right).$$

In seguito vediamo delle regole importanti delle derivate parziali, che sono analoghe a quelle viste in una dimensione.

Proposizione 2.3.8 (Proprietà algebriche delle derivate parziali)**.** *Siano $f, g : \Omega \to \mathbb{R}$, $\mathbf{x}_0 \in \Omega$ e $j \in \{1, \ldots, n\}$. Se esistono $\frac{\partial f}{\partial x_j}(\mathbf{x}_0)$ e $\frac{\partial g}{\partial x_j}(\mathbf{x}_0)$, allora:*

1. ***Linearità**:*
$$\frac{\partial(\lambda f + \mu g)}{\partial x_j}(\mathbf{x}_0) = \lambda \frac{\partial f}{\partial x_j}(\mathbf{x}_0) + \mu \frac{\partial g}{\partial x_j}(\mathbf{x}_0) \quad \forall \lambda, \mu \in \mathbb{R};$$

2. ***Regola del prodotto** (se f e g sono continue in $\mathbf{x}_0$):*
$$\frac{\partial(f \cdot g)}{\partial x_j}(\mathbf{x}_0) = \frac{\partial f}{\partial x_j}(\mathbf{x}_0) \cdot g(\mathbf{x}_0) + f(\mathbf{x}_0) \cdot \frac{\partial g}{\partial x_j}(\mathbf{x}_0);$$

3. ***Regola del quoziente** (se f e g sono continue in $\mathbf{x}_0$ e $g(\mathbf{x}_0) \neq 0$):*
$$\frac{\partial}{\partial x_j}\left(\frac{f}{g}\right)(\mathbf{x}_0) = \frac{\frac{\partial f}{\partial x_j}(\mathbf{x}_0) \cdot g(\mathbf{x}_0) - f(\mathbf{x}_0) \cdot \frac{\partial g}{\partial x_j}(\mathbf{x}_0)}{(g(\mathbf{x}_0))^2}.$$

Derivata direzionale

Consideriamo una funzione $f : \Omega \subseteq \mathbb{R}^2 \to \mathbb{R}$ in 2 variabili $f(x, y)$, che rappresenta una superficie nello spazio tridimensionale:

$$\{(x, y, z) \in \mathbb{R}^3 : z = f(x, y), (x, y) \in \Omega\}.$$

Abbiamo definito le derivate parziali in un punto $\mathbf{x}_0 = (x_0, y_0) \in \Omega$ come:

- $\frac{\partial f}{\partial x}(\mathbf{x}_0)$, che misura il tasso di variazione di f rispetto alla variabile x mantenendo $y = y_0$ costante;
- $\frac{\partial f}{\partial y}(\mathbf{x}_0)$ che ci dice la stessa cosa solo che stavolta analizziamo il tasso di variazione di f rispetto y e teniamo fisso $x = x_0$.

Queste derivate parziali descrivono come la funzione varia lungo le direzioni degli assi coordinati. Tuttavia, se vogliamo studiare la variazione della funzione in una direzione arbitraria, dobbiamo estendere il concetto alle derivate direzionali.

Definizione 2.3.9 (**Derivata direzionale**)**.** Siano $f : \Omega \to \mathbb{R}$, $\mathbf{x}_0 \in \Omega$ e $\mathbf{v} \in \mathbb{R}^n$ un vettore non nullo. La derivata direzionale di f in $\mathbf{x}_0$ lungo la direzione $\mathbf{v}$ è il limite

$$\partial_{\mathbf{v}} f(\mathbf{x}_0) = \lim_{h \to 0} \frac{f(\mathbf{x}_0 + h\mathbf{v}) - f(\mathbf{x}_0)}{h},$$

se esiste finito.

Osservazione 2.3.10. Le derivate parziali sono casi particolari di derivate direzionali:

$$\partial_{\mathbf{e}_j} f(\mathbf{x}_0) = \frac{\partial f}{\partial x_j}(\mathbf{x}_0).$$

Enunciamo adesso un risultato fondamentale sulle derivate direzionali, che dimostreremo nella Sezione 2.6.

Teorema 2.3.11 (**Relazione tra derivata direzionale e gradiente**)**.** *Sia $f \in C^1(\Omega)$. Per ogni punto $\mathbf{x}_0 \in \Omega$ e per ogni vettore $\mathbf{v} \in \mathbb{R}^n$ con $\mathbf{v} \neq \mathbf{0}$, la derivata direzionale di f in $\mathbf{x}_0$ lungo $\mathbf{v}$ è data da*

$$\partial_{\mathbf{v}} f(\mathbf{x}_0) = \nabla f(\mathbf{x}_0) \cdot \mathbf{v}.$$

Osservazione 2.3.12. Notiamo che, se $\|\mathbf{v}\| = 1$, la derivata direzionale rappresenta il tasso di variazione di f nella direzione $\mathbf{v}$.

Differenziabilità

L'idea di differenziabilità estende al caso multidimensionale il concetto di avere un spazio affine tangente che approssima bene la funzione.

Se torniamo al caso monodimensionale, una funzione $f : \mathbb{R} \to \mathbb{R}$, si dice derivabile in un punto $x_0 \in \mathbb{R}$ se ha una retta tangente in quel punto, che è data da

$$y = f(x_0) + f'(x_0)(x - x_0),$$

e che quindi approssima $f(x)$ con un errore che tende a zero più rapidamente di $|x - x_0|$.

Possiamo generalizzare questa idea: una funzione $f : \mathbb{R}^n \to \mathbb{R}$ è differenziabile in $\mathbf{x}_0$ se esiste un iperpiano tangente (l'analogo n-dimensionale della retta tangente) che la approssima bene.

Definizione 2.3.13 (**Differenziabilità**)**.** Sia $f : \Omega \to \mathbb{R}$. Si dice che f è differenziabile in $\mathbf{x}_0 \in \Omega$ se esiste una trasformazione lineare $L : \mathbb{R}^n \to \mathbb{R}$ tale che:

$$\lim_{\mathbf{h}\to\mathbf{0}} \frac{f(\mathbf{x}_0 + \mathbf{h}) - f(\mathbf{x}_0) - L(\mathbf{x}_0)\mathbf{h}}{\|\mathbf{h}\|} = 0.$$

Questa L è detta differenziale di f in $\mathbf{x}_0$.

Osservazione 2.3.14 (**Interpretazione geometrica**)**.** La differenziabilità significa che, vicino a $\mathbf{x}_0$, f è ben approssimata da una funzione affine:

$$f(\mathbf{x}_0 + \mathbf{h}) \approx f(\mathbf{x}_0) + L(\mathbf{x}_0)\mathbf{h}.$$

Il grafico ammette quindi un iperpiano tangente in $\mathbf{x}_0$. Per $n = 1$ differenziabilità e derivabilità coincidono.

Teorema 2.3.15 (**Relazione fondamentale**)**.** *Sia $f : \Omega \to \mathbb{R}$ e $\mathbf{x}_0 \in \Omega$.*

1. *Se f è differenziabile in $\mathbf{x}_0$, allora*
   - *f è continua in $\mathbf{x}_0$;*
   - *esistono tutte le derivate parziali in $\mathbf{x}_0$;*
   - *$L(\mathbf{x}_0)\mathbf{h} = \nabla f(\mathbf{x}_0) \cdot \mathbf{h}$.*
2. *L'esistenza di tutte le derivate parziali* non *implica la differenziabilità.*
3. *Se tutte le derivate parziali esistono in un in $B(\mathbf{x}_0, r)$ per qualche $r > 0$ e sono continue in $\mathbf{x}_0$, allora f è differenziabile in $\mathbf{x}_0$.*

Osservazione 2.3.16. La continuità delle derivate parziali è condizione sufficiente ma non necessaria: esistono funzioni differenziabili con derivate parziali non continue.

Osservazione 2.3.17. Per funzioni in una variabile ($n = 1$), i concetti di derivabilità e differenziabilità coincidono.

Definizione 2.3.18 (**Spazio $C^1(\Omega)$**)**.** Una funzione $f : \Omega \to \mathbb{R}$ è di classe $C^1(\Omega)$ se tutte le sue derivate parziali prime esistono e sono continue su Ω. Denotiamo con $C^1(\Omega)$ l'insieme di tutte le funzioni con derivate parziali prime continue su Ω.

Derivate parziali seconde

La derivata parziale di una funzione in n variabili rimane sempre una funzione in n variabili. Possiamo quindi, analogamente al caso monodimensionale, derivare nuovamente una derivata parziale, ottenendo in questo modo una derivata parziale di ordine superiore, anche se, per i nostri scopi, è sufficiente arrivare al secondo ordine.

Definizione 2.3.19 (**Derivate parziali seconde**)**.** Siano $f \in C^1(\Omega)$ e $\mathbf{x}_0 \in \Omega$. Se per ogni $i, j \in \{1, \dots n\}$ esistono le derivate parziali delle derivate parziali prime in $\mathbf{x}_0$, , cioè:

$$\frac{\partial}{\partial x_j}\left(\frac{\partial f}{\partial x_i}\right)(\mathbf{x}_0),$$

allora queste sono le derivate parziali seconde di f in $\mathbf{x}_0$, indicate con:

$$\frac{\partial^2 f}{\partial x_j \partial x_i}(\mathbf{x}_0) \quad \text{per } i \neq j, \qquad \frac{\partial^2 f}{\partial x_i^2}(\mathbf{x}_0) \quad \text{per } i = j.$$

Definizione 2.3.20 (**Spazio** $C^2(\Omega)$)**.** Una funzione $f : \Omega \to \mathbb{R}$ è di classe $C^2(\Omega)$ se tutte le sue derivate parziali prime e seconde esistono e sono continue su Ω. Denotiamo con $C^2(\Omega)$ l'insieme di tutte le funzioni con derivate parziali prime e seconde continue su Ω.

Lemma 2.3.21 (**Lemma di Schwarz**)**.** *Se $f \in C^2(\Omega)$, allora, per ogni $\mathbf{x} \in \Omega$ e per ogni $i, j \in \{1, \dots, n\}$ vale*

$$\frac{\partial^2 f}{\partial x_j \partial x_i}(\mathbf{x}) \;=\; \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}).$$

Si possono raccogliere tutte le derivate parziali seconde di f nel punto $\mathbf{x} \in \Omega$ in una matrice quadrata $(n \times n)$ chiamata matrice Hessiana.

Definizione 2.3.22 (**Matrice Hessiana**)**.** Sia $f \in C^2(\Omega)$. La matrice Hessiana di f in $\mathbf{x} \in \Omega$ è la matrice $n \times n$ delle derivate parziali seconde:

$$\mathbf{H}f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(\mathbf{x}) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_2^2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_n \partial x_2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(\mathbf{x}) \end{bmatrix}.$$

Osservazione 2.3.23. Per il Lemma di Schwarz, la matrice Hessiana di una funzione C^2 è simmetrica.

2.4 Curve e funzioni vettoriali

Definizione 2.4.1 (**Curva**)**.** Una curva in $\Omega \subseteq \mathbb{R}^n$ è una funzione continua $\gamma : I \to \Omega$ definita su un intervallo $I \subseteq \mathbb{R}$. Indichiamo con γ_j la j-esima componente di γ, ovvero

$$\gamma(t) = \Big(\gamma_1(t), \gamma_2(t), \dots, \gamma_n(t)\Big) \quad \forall\, t \in I.$$

Se tutte le componenti sono differenziabili su I, allora la curva è detta differenziabile su I, e definiamo la derivata di γ come

$$\gamma'(t) = \Big(\gamma_1'(t), \gamma_2'(t), \dots, \gamma_n'(t)\Big) \in \mathbb{R}^n \quad \forall\, t \in I,$$

dove $\gamma_j'(t)$ indica la derivata usuale della funzione reale γ_j rispetto a t.

Definizione 2.4.2 (**Funzioni vettoriali di più variabili**)**.** Siano $m, n \in \mathbb{N}$ e $\Omega \subseteq \mathbb{R}^n$. Se $F : \Omega \to \mathbb{R}^m$ è data da

$$F(\mathbf{x}) = \begin{pmatrix} F_1(\mathbf{x}) \\ \vdots \\ F_m(\mathbf{x}) \end{pmatrix},$$

dove $F_j : \Omega \to \mathbb{R}$ sono funzioni in più variabili, allora F è una funzione vettoriale di più variabili. Se $m = n$, allora F è un campo vettoriale.

Osservazione 2.4.3. Le curve sono funzioni vettoriali continue, corrispondenti al caso $n = 1$ e $m \geq 1$, cioè funzioni $\gamma : I \subset \mathbb{R} \to \mathbb{R}^m$ continue.

Definizione 2.4.4 (**Matrice Jacobiana**)**.** Siano $F : \Omega \to \mathbb{R}^m$ e $\mathbf{x} \in \Omega$. Se $F_1, \dots, F_m$ ammettono derivate parziali in $\mathbf{x}$, allora la matrice $m \times n$ avente per righe i gradienti in $\mathbf{x}$ delle funzioni F_j per $j \in \{1, \dots, m\}$ è la matrice Jacobiana di F in $\mathbf{x}$:

$$\mathbf{J}F(\mathbf{x}) = \begin{bmatrix} \nabla F_1(\mathbf{x}) \\ \nabla F_2(\mathbf{x}) \\ \vdots \\ \nabla F_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1}(\mathbf{x}) & \frac{\partial F_1}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial F_1}{\partial x_n}(\mathbf{x}) \\ \frac{\partial F_2}{\partial x_1}(\mathbf{x}) & \frac{\partial F_2}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial F_2}{\partial x_n}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1}(\mathbf{x}) & \frac{\partial F_m}{\partial x_2}(\mathbf{x}) & \cdots & \frac{\partial F_m}{\partial x_n}(\mathbf{x}) \end{bmatrix}.$$

Osservazione 2.4.5. Si può osservare che, mentre ogni riga della matrice Jacobiana rappresenta il gradiente di una componente della funzione vettoriale F, le colonne della Jacobiana contengono le derivate parziali della funzione rispetto alle singole variabili indipendenti. In altre parole, la j-esima colonna di $\mathbf{J}F$ è il vettore delle derivate parziali di tutte le componenti di F rispetto alla variabile x_j.

2.5 Regola della catena

La regola della catena è uno dei risultati fondamentali del calcolo differenziale: essa descrive come calcolare la derivata di una funzione composta, permettendo di "trasferire" il tasso di variazione dall'interno verso l'esterno della composizione.

Proposizione 2.5.1 (**Caso monodimensionale**). *Sia $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una funzione derivabile e sia $g : \Omega \subseteq \mathbb{R} \to I$ una funzione derivabile. La funzione composta $f \circ g : \Omega \to \mathbb{R}$ è allora derivabile e vale*

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

Proposizione 2.5.2 (**Regola della catena per funzioni in più variabili composte con curve**). *Sia $f \in C^1(\Omega)$ e sia $\gamma : I \subseteq \mathbb{R} \to \Omega$ una curva differenziabile. Consideriamo la funzione composta $F : I \to \mathbb{R}$ come $F(t) = f(\gamma(t))$. Allora F è derivabile in ogni punto $t \in I$ e si ha*

$$\frac{d}{dt} f(\gamma(t)) = \nabla f(\gamma(t)) \cdot \gamma'(t) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\gamma(t)) \frac{d\gamma_i}{dt}(t).$$

Osservazione 2.5.3. La derivata della funzione composta $f(\gamma(t))$ rappresenta il tasso di variazione di f lungo la traiettoria $\gamma(t)$. Equivalentemente, essa coincide con la derivata direzionale di f nel punto $\gamma(t)$ lungo la direzione $\gamma'(t)$.

Proposizione 2.5.4 (**Regola della catena per funzioni in una variabile composte con funzioni in più variabili**). *Sia $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$ una funzione differenziabile in $\mathbf{x}_0 \in \Omega$, e sia $\phi : I \subseteq \mathbb{R} \to \mathbb{R}$ una funzione derivabile, con $f(\Omega) \subseteq I$. Allora $\phi \circ f : \Omega \to \mathbb{R}$ è differenziabile in $\mathbf{x}_0$ e vale*

$$\frac{\partial(\phi \circ f)}{\partial x_j}(\mathbf{x}_0) = \phi'(f(\mathbf{x}_0)) \frac{\partial f}{\partial x_j}(\mathbf{x}_0), \qquad j = 1, \ldots, n,$$

cioè in forma compatta

$$\nabla(\phi \circ f)(\mathbf{x}_0) = \phi'(f(\mathbf{x}_0)) \nabla f(\mathbf{x}_0).$$

I casi precedenti sono esempi particolari di una formulazione più generale.

Proposizione 2.5.5 (**Regola della catena matriciale**). *Se $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^m$ è differenziabile in $\mathbf{x}_0 \in \Omega$ e $\phi : \Theta \subseteq \mathbb{R}^m \to \mathbb{R}^k$ è differenziabile in $f(\mathbf{x}_0) \in \Theta$, con $f(\Omega) \subseteq \Theta$, allora*

$$J(\phi \circ f)(\mathbf{x}_0) = J\phi(f(\mathbf{x}_0)) Jf(\mathbf{x}_0),$$

dove Jf e $J\phi$ denotano le matrici Jacobiane.

2.6 Sviluppo di Taylor

Uno degli strumenti più potenti per studiare il comportamento locale di una funzione è lo sviluppo di Taylor. L'idea di base è quella di approssimare una funzione, potenzialmente complicata, vicino a un punto fissato, mediante un polinomio costruito a partire dalle sue derivate calcolate in tale punto.

Iniziamo richiamando la teoria nel caso unidimensionale, per poi estenderla alle funzioni di più variabili.

Definizione 2.6.1 (**Polinomio di Taylor in una variabile**)**.** Sia $f : I \to \mathbb{R}$ una funzione derivabile n volte in un punto $x_0 \in I$. Il polinomio di Taylor di grado n di f centrato in x_0 è definito da

$$T_{n,x_0} f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k.$$

Il polinomio di Taylor approssima localmente la funzione. La precisione di tale approssimazione è descritta dal termine di resto.

2.6.1 Sviluppo di Taylor con resto di Peano

Teorema 2.6.2 (**Sviluppo di Taylor in una variabile con resto di Peano**)**.** *Sia $I \subseteq \mathbb{R}$ un intervallo e sia $f : I \to \mathbb{R}$ una funzione derivabile n volte in un punto $x_0 \in I$, con $n \geq 1$. Allora, per $x \to x_0$, lo sviluppo di Taylor di ordine n con resto di Peano è dato da*

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + o\left((x - x_0)^n\right),$$

ovvero, in forma compatta

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + o\Big((x - x_0)^n\Big).$$

Notazione 2.6.3. Il simbolo $o\left(g(x)\right)$ indica un infinitesimo di ordine superiore rispetto a $g(x)$ per $x \to x_0$. Quindi è una quantità trascurabile rispetto a $g(x)$ quando ci avviciniamo a x_0. Nel caso scalare, si dice che $f(x) = o(g(x))$ per $x \to x_0$ se

$$\lim_{x \to x_0} \frac{f(x)}{g(x)} = 0.$$

Nel caso di funzioni con più variabili, non potendo dividere per $(\mathbf{x} - \mathbf{x}_0)$, si utilizza $o\Big(\|\mathbf{x} - \mathbf{x}_0\|\Big)$, che va a zero più velocemente della distanza $\|\mathbf{x} - \mathbf{x}_0\|$.
Per funzioni vettoriali, si scrive $\mathbf{o}\Big(\|\mathbf{x} - \mathbf{x}_0\|\Big)$, che è una sorta di o-piccolo vettoriale, e che significa che ogni componente dell'errore vettoriale è $o\Big(\|\mathbf{x} - \mathbf{x}_0\|\Big)$

Osservazione 2.6.4 (**Approssimazione lineare**)**.** Per $n = 1$ si ottiene lo sviluppo di Taylor al primo ordine:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0),$$

dove il polinomio al secondo membro rappresenta l'equazione della retta tangente al grafico di f nel punto $(x_0, f(x_0))$.

Il principio dello sviluppo di Taylor si estende alle funzioni in più variabili, con la differenza che il comportamento locale deve tenere conto di tutte le direzioni di variazione della funzione. In questo contesto, le derivate ordinarie vengono sostituite da derivate parziali, e l'approssimazione lineare è data dall'iperpiano tangente al grafico della funzione.

Teorema 2.6.5 (**Sviluppo di Taylor al primo ordine in più variabili**)**.** *Sia $f \in C^1(\Omega)$ con $\Omega \subseteq \mathbb{R}^n$. Allora, per ogni $\mathbf{x}_0 \in \Omega$ e per $\mathbf{x} \to \mathbf{x}_0$, vale*

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) + o\Big(\|\mathbf{x} - \mathbf{x}_0\|\Big),$$

ovvero, in forma esplicita:

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\mathbf{x}_0)(x_i - a_i) + o\Big(\|\mathbf{x} - \mathbf{x}_0\|\Big).$$

Teorema 2.6.6 (**Sviluppo di Taylor al secondo ordine con resto di Peano**). *Sia $f \in C^2(\Omega)$ con $\Omega \subseteq \mathbb{R}^n$, e sia $\mathbf{x}_0 \in \Omega$. Allora, per $\mathbf{x} \to \mathbf{x}_0$, vale*

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^\top H_f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + o(\|\mathbf{x} - \mathbf{x}_0\|^2).$$

Possiamo usare la formula di Taylor per dimostrare il Teorema 2.3.11.

Dimostrazione. [**del Teorema 2.3.11**] Per definizione, la derivata direzionale è:

$$\partial_v f(\mathbf{x}) = \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{v}) - f(\mathbf{x})}{h}.$$

Dato che f è differenziabile in $\mathbf{x}$, possiamo scrivere lo sviluppo di Taylor al primo ordine in $\mathbf{x} \in \Omega$ (Teorema 2.6.5):

$$f(\mathbf{x} + h\mathbf{v}) = f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot (h\mathbf{v}) + o(h).$$

Sostituendo nella definizione di derivata direzionale otteniamo:

$$\begin{aligned}
\partial_v f(\mathbf{x}) &= \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{v}) - f(\mathbf{x})}{h} \\
&= \lim_{h \to 0} \frac{f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot (h\mathbf{v}) + o(h) - f(\mathbf{x})}{h} \\
&= \lim_{h \to 0} \frac{h\nabla f(\mathbf{x}) \cdot \mathbf{v} + o(h)}{h} = \lim_{h \to 0} \left(\nabla f(\mathbf{x}) \cdot \mathbf{v} + \frac{o(h)}{h} \right).
\end{aligned}$$

Dato che il limite è un'operazione lineare, concludiamo che

$$\partial_v f(\mathbf{x}) = \nabla f(\mathbf{x}) \cdot \mathbf{v} + \lim_{h \to 0} \left(\frac{o(h)}{h} \right) = \nabla f(\mathbf{x}) \cdot \mathbf{v}.$$

□

Teorema 2.6.7 (**Sviluppo di Taylor del primo ordine per funzioni vettoriali**). *Sia $F \in C^1(\Omega; \mathbb{R}^m)$ con $\Omega \subseteq \mathbb{R}^n$ e sia $\mathbf{x}_0 \in \Omega$. Allora, per $\mathbf{x} \to \mathbf{x}_0$,*

$$F(\mathbf{x}) = F(\mathbf{x}_0) + JF(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + \mathbf{o}\Big(\|\mathbf{x} - \mathbf{x}_0\|\Big).$$

2.6.2 Sviluppo di Taylor con resto di Lagrange

Il resto di Peano fornisce solo un'indicazione asintotica dell'errore. Esiste però una forma più precisa, dovuta a Lagrange, che permette di esprimere esattamente il resto tramite una derivata di ordine superiore.

Il risultato che permette di dimostrare la formula del resto di Lagrange è il teorema del valore medio differenziale.

Teorema 2.6.8 (**Teorema di Lagrange**). *Sia $f : [a, b] \to \mathbb{R}$ una funzione continua su $[a, b]$ e derivabile su (a, b). Allora esiste almeno un punto $c \in (a, b)$ tale che*

$$f'(c) = \frac{f(b) - f(a)}{b - a}.$$

Osservazione 2.6.9. Il teorema di Lagrange afferma che, su un intervallo, la pendenza della retta secante tra due punti coincide con la derivata della funzione in almeno un punto intermedio. Questo principio è alla base della dimostrazione della formula dello sviluppo di Taylor con resto di Lagrange.

Teorema 2.6.10 (**Sviluppo di Taylor con resto di Lagrange**). *Sia $f : I \subseteq \mathbb{R} \to \mathbb{R}$ una funzione $n+1$ volte derivabile nell'intervallo I e siano $x, x_0 \in I$. Allora esiste un punto θ compreso tra x_0 e x tale che*

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k + \frac{f^{(n+1)}(\theta)}{(n+1)!}(x - x_0)^{n+1}.$$

Osservazione 2.6.11. Il termine

$$R_{n+1}(x) = \frac{f^{(n+1)}(\theta)}{(n+1)!}(x - x_0)^{n+1}$$

rappresenta il resto dello sviluppo e fornisce una stima esplicita dell'errore nell'approssimazione di $f(x)$ tramite il polinomio di Taylor di grado n.

Questo procedimento si può estendere anche a funzioni di più variabili. Prima di fare ciò, introduciamo la seguente notazione.

Notazione 2.6.12 (**Segmento**). Sia $\Omega \subseteq \mathbb{R}^n$, e due punti distinti $\mathbf{x}, \mathbf{y} \in \Omega$, si definisce segmento di estremi $\mathbf{x}$ e $\mathbf{y}$ l'insieme

$$[\mathbf{x}, \mathbf{y}] = \{(1-t)\mathbf{x} + t\mathbf{y} : t \in [0,1]\}.$$

Possiamo rappresentare un punto $\mathbf{z}$ sul segmento di estremi $\mathbf{x}$ e $\mathbf{y}$ anche come

$$\mathbf{z} = \mathbf{x} + t(\mathbf{y} - \mathbf{x}).$$

Teorema 2.6.13 (**Teorema del valore medio in n-dimensioni**). *Sia $f \in C^1(\Omega)$. Per ogni $\mathbf{x}, \mathbf{y} \in \Omega$, tali che $[\mathbf{x}, \mathbf{y}] \subset \Omega$, esiste un punto*

$$\mathbf{z} = \mathbf{x} + t(\mathbf{y} - \mathbf{x}), \quad \textit{per qualche} \quad t \in (0,1),$$

tale che

$$f(\mathbf{y}) - f(\mathbf{x}) = \nabla f(\mathbf{z}) \cdot (\mathbf{y} - \mathbf{x}).$$

Dimostrazione. Consideriamo la funzione ausiliaria $\phi : [0,1] \to \mathbb{R}$ definita da:

$$\phi(t) = f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})).$$

La funzione ϕ è differenziabile in $[0,1]$ per la regola della catena, poiché f è differenziabile in Ω e il segmento $[\mathbf{x}, \mathbf{y}]$ è contenuto in Ω. Calcoliamo la derivata di ϕ:

$$\phi'(t) = \nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) \cdot (\mathbf{y} - \mathbf{x}).$$

Applicando il teorema di Lagrange in una variabile a ϕ sull'intervallo $[0,1]$, esiste $t_0 \in (0,1)$ tale che:

$$\phi(1) - \phi(0) = \phi'(t_0)(1 - 0).$$

Sostituendo le espressioni di ϕ e ϕ', troviamo

$$f(\mathbf{y}) - f(\mathbf{x}) = \nabla f(\mathbf{x} + t_0(\mathbf{y} - \mathbf{x})) \cdot (\mathbf{y} - \mathbf{x}).$$

Ponendo $\mathbf{z} = \mathbf{x} + t_0(\mathbf{y} - \mathbf{x})$, otteniamo la tesi. □

Grazie a questo risultato, il resto può essere espresso in forma analoga al caso monodimensionale, e lo sviluppo di Taylor al primo ordine con il resto assume la forma seguente.

Teorema 2.6.14 (**Taylor al primo ordine con resto di Lagrange in $\mathbb{R}^n$**). *Sia $f \in C^1(\Omega)$ e siano $\mathbf{x}_0, \mathbf{x} \in \Omega$ tali che $[\mathbf{x}_0, \mathbf{x}] \subset \Omega$. Allora esiste $\xi \in [\mathbf{x}_0, \mathbf{x}]$ tale che*

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\xi) \cdot (\mathbf{x} - \mathbf{x}_0).$$

L'accuratezza dell'approssimazione migliora aggiungendo termini di ordine superiore, che catturano anche la curvatura locale della funzione.

Teorema 2.6.15 (**Taylor al secondo ordine con resto di Lagrange in $\mathbb{R}^n$**). *Sia $f \in C^2(\Omega)$ e siano $\mathbf{x}_0, \mathbf{x} \in \Omega$ tali che $[\mathbf{x}_0, \mathbf{x}] \subset \Omega$. Allora esiste $\xi \in [\mathbf{x}_0, \mathbf{x}]$ tale che*

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^\top \mathbf{H} f(\xi)(\mathbf{x} - \mathbf{x}_0). \tag{2.6.1}$$

Dimostrazione. Riduciamo il problema in più variabili a una variabile lavorando lungo il segmento che unisce i due punti $\mathbf{x}$ e $\mathbf{y}$.
Definiamo

$$\phi(t) := f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}), \quad \forall\, t \in [0, 1],$$

e osserviamo che, dato che $f \in C^2(\Omega)$, anche $\phi \in C^2([0, 1])$.
Ora applichiamo la formula dello sviluppo di Taylor del secondo ordine con resto di Lagrange, che ci dice che esiste $\theta \in (0, 1)$ tale che

$$\phi(1) = \phi(0) + \frac{d}{dt}\phi(0)(1 - 0) + \frac{1}{2}\frac{d^2}{dt^2}\phi(\theta)(1 - 0)^2.$$

La derivata prima è

$$\frac{d}{dt}\phi(t) = \nabla f\Big(\mathbf{x} + t(\mathbf{y} - \mathbf{x})\Big)(\mathbf{y} - \mathbf{x}),$$

in particolare

$$\frac{d}{dt}\phi(0) = \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}).$$

La derivata seconda invece è

$$\frac{d^2}{dt^2}\phi(t) = (\mathbf{y} - \mathbf{x})^\top \mathbf{H} f\Big(\mathbf{x} + t(\mathbf{y} - \mathbf{x})\Big)(\mathbf{y} - \mathbf{x}).$$

Otteniamo quindi

$$\phi(1) = f(x) + \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}) + \frac{1}{2}(\mathbf{y} - \mathbf{x})^\top \mathbf{H} f\Big(\mathbf{x} + \theta(\mathbf{y} - \mathbf{x})\Big)(\mathbf{y} - \mathbf{x}).$$

Ponendo

$$\xi := \mathbf{x} + \theta(\mathbf{y} - \mathbf{x}) \in \Omega,$$

otteniamo (2.6.1). □

Capitolo 3

Ottimizzazione Convessa

3.1 Introduzione all'ottimizzazione

L'ottimizzazione consiste nel massimizzare o minimizzare una funzione reale scegliendo in modo sistematico i valori delle variabili decisionali da un insieme di ingresso, al fine di determinare il corrispondente valore ottimale in uscita. In altre parole, studia metodi per trovare i valori delle variabili decisionali che minimizzano (o massimizzano) una funzione obiettivo, eventualmente soggetta a vincoli.

Definizione 3.1.1 (**Problema di ottimizzazione**)**.** Siano f una funzione scalare $f : \mathbb{R}^n \to \mathbb{R}$ e Ω un sottoinsieme di $\mathbb{R}^n$. f è una funzione di variabile $\mathbf{x} \in \mathbb{R}^n$, detto anche vettore delle variabili decisionali. Allora, un problema di ottimizzazione è un problema della forma:

$$\min_{\mathbf{x}\in\Omega} f(\mathbf{x})$$

L'abbreviazione *min* sta per "minimizzare". La funzione $f(\mathbf{x})$ è denominata funzione obiettivo e l'insieme Ω è l'insieme dei vincoli, che spesso ha la forma:

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n \mid g_i(\mathbf{x}) \geq 0;\ i = 1, \ldots, m; \quad \text{e} \quad h_j(\mathbf{x}) = 0;\ j = 1, \ldots, p\},$$

dove g_i sono i vincoli di disuguaglianza e h_j sono i vincoli di uguaglianza ed entrambi sono funzioni $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$.

Osservazione 3.1.2. Un problema di massimizzazione può essere sempre riformulato in modo equivalente come un problema di minimizzazione. In particolare, massimizzare una funzione $f(x)$ equivale a minimizzare il suo opposto:

$$\max f(\mathbf{x}) = -\min[-f(\mathbf{x})].$$

3.1.1 Tipi di soluzioni

Definizione 3.1.3 (**Soluzione ottima**)**.** La soluzione ottima $\bar{\mathbf{x}}$ è un punto in Ω in cui la funzione obiettivo raggiunge il valore minimo:

$$f(\bar{\mathbf{x}}) \leq f(\mathbf{x}) \qquad \forall \mathbf{x} \in \Omega.$$

Questa soluzione può non esistere o non essere unica.

Definizione 3.1.4 (**Valore ottimo**)**.** È il valore numerico della funzione obiettivo calcolato nella soluzione ottima, ovvero il valore minimo nell'insieme Ω:

$$f^* = f(\bar{\mathbf{x}}) = \min_{x\in\Omega} f(\mathbf{x}).$$

Osservazione 3.1.5. Il valore ottimo è uno scalare: $f^* \in \mathbb{R}$.

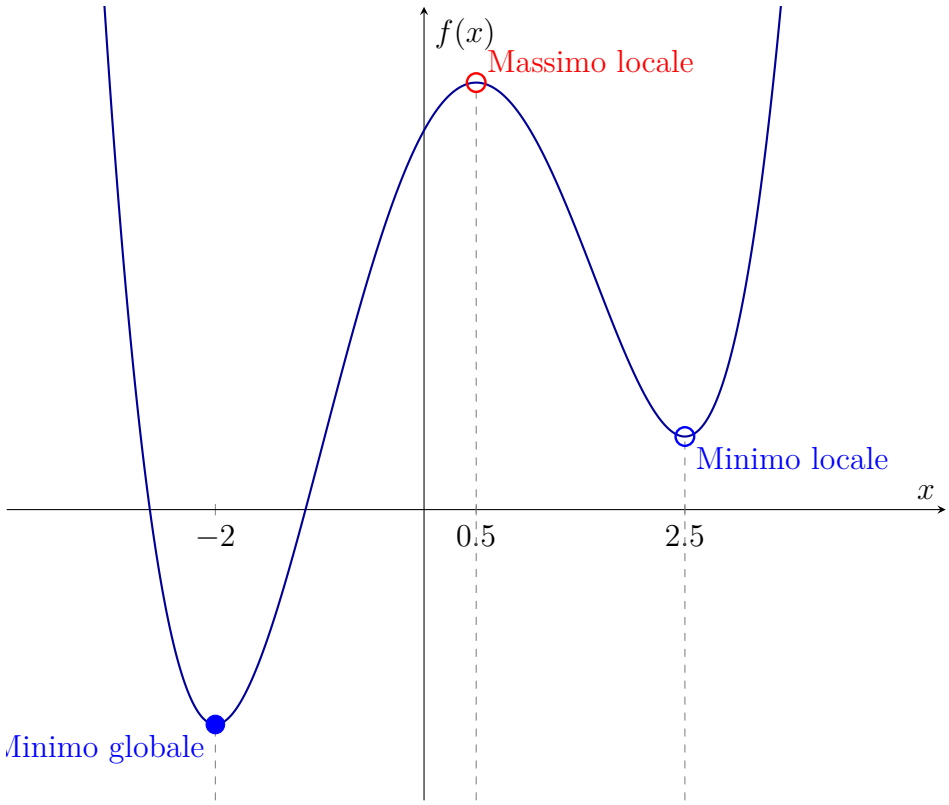


Figura 3.1: Schema di massimo e minimo globale e locale di una funzione

3.1.2 Minimo locale e globale

Se abbiamo un problema di minimizzazione come rappresentato nella sezione 3.1.1, allora un punto che rappresenti una soluzione ottima $\bar{\mathbf{x}}$ viene chiamato minimo globale.

Definizione 3.1.6 (**Minimo globale**)**.** Sia $f(\bar{\mathbf{x}})$ il valore più basso che f può assumere in tutto il dominio ammissibile, ovvero,

$$f(\bar{\mathbf{x}}) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \in \Omega,$$

allora $\bar{\mathbf{x}}$ è un minimo globale di f in Ω.

Definizione 3.1.7 (**Minimo locale**)**.** Diciamo che $\bar{\mathbf{x}} \in \Omega$ è un punto di minimo locale per f se esiste $\varepsilon > 0$ tale che

$$f(\bar{\mathbf{x}}) \leq f(\mathbf{x}) \qquad \forall\ \mathbf{x} \in B(\bar{\mathbf{x}}, \varepsilon)\ \cap\ \Omega.$$

La Figura 3.1 rappresenta i concetti sopra elencati.

Un minimo locale o globale non è necessariamente unico. Però, se rendiamo le disuguaglianze strette, otteniamo una definizione che implica l'unicità.

Definizione 3.1.8 (**Minimo globale stretto**)**.** Diciamo che $\bar{\mathbf{x}} \in \Omega$ è un punto di minimo globale stretto per f se esiste $\varepsilon > 0$ tale che

$$f(\bar{\mathbf{x}}) < f(\mathbf{x}) \quad \forall \mathbf{x} \in \Omega \backslash \{\bar{\mathbf{x}}\}.$$

Definizione 3.1.9 (**Minimo locale stretto**)**.** Diciamo che $\bar{\mathbf{x}} \in \Omega$ è un punto di minimo locale stretto per f se esiste $\varepsilon > 0$ tale che

$$f(\bar{\mathbf{x}}) < f(\mathbf{x}) \qquad \forall\ \mathbf{x} \in B(\bar{\mathbf{x}}, \varepsilon)\ \cap\ \Omega\ \backslash \{\bar{\mathbf{x}}\}.$$

Osservazione 3.1.10. Se un minimo globale è stretto, allora esso è unico in tutto il dominio. Analogamente, se un minimo locale è stretto, esso è unico nell'intersezione della palla $B(\bar{\mathbf{x}}, \varepsilon)$ con il dominio, ma non necessariamente in tutto il dominio.

Osservazione 3.1.11. Un minimo globale è anche un minimo locale; però, come vedremo in seguito, è molto più facile trovare un minimo locale.

3.1.3 Problemi senza vincoli

In questa sezione vediamo le condizioni di ottimalità, nel contesto dei problemi senza vincoli (unconstrained). Un problema di ottimizzazione senza vincoli consiste nel determinare il punto qualsiasi $\bar{\mathbf{x}} \in \mathbb{R}^n$ che minimizza una funzione obiettivo reale $f : \mathbb{R}^n \to \mathbb{R}$, senza alcuna restrizione esplicita sull'insieme delle soluzioni ammissibili.

Osservazione 3.1.12. Un problema di ottimizzazione senza vincoli ha la forma standard

$$\min f(\mathbf{x}),$$

senza specificare condizioni sulla variabile $\mathbf{x}$.

Come facciamo però a sapere se un punto è un minimo?

Condizioni di ottimalità

Supponiamo da questo punto fino alla fine del capitolo che $\Omega \subseteq \mathbb{R}^n$ è un insieme aperto.

Teorema 3.1.13 (**Teorema di Fermat**). *Siano $f \in C^1(\Omega)$ e $\mathbf{x}_0 \in \Omega$. Se $\mathbf{x}_0$ è un punto di massimo o di minimo locale per f, allora*

$$\nabla f(\mathbf{x}_0) = 0,$$

ovvero:

$$\frac{\partial f}{\partial x_j}(\mathbf{x}_0) = 0 \quad \forall\ j \in \{1, \dots, n\}.$$

Definizione 3.1.14 (**Punto critico**). Siano $f \in C^1(\Omega)$ e $\mathbf{x}_0 \in \Omega$. $\mathbf{x}_0$ è un punto critico di f se $\nabla f(\mathbf{x}_0) = 0$.

Definizione 3.1.15 (**Punto sella**). Un punto $\mathbf{x}_0 \in \Omega$ è detto punto di sella per f se x_0 è un punto critico che non è né un punto di massimo locale né di minimo locale.

Osservazione 3.1.16. Se $\mathbf{x}_0$ è un punto di sella per f, allora in ogni palla $B(\mathbf{x}_0, r)$ la funzione f assume sia valori maggiori sia minori di $f(\mathbf{x}_0)$.

Per poter dimostrare il teorema di Fermat dobbiamo prima spiegare il concetto di direzione di discesa.

Definizione 3.1.17 (**Direzione di discesa**). Siano $f \in C^1(\Omega)$ e $\mathbf{x}_0 \in \Omega$. Un vettore $\mathbf{d} \in \mathbb{R}^n$ è una direzione di discesa di f in $\mathbf{x}_0$ se esiste $\bar{\alpha} > 0$ tale che

$$f(\mathbf{x}_0 + \alpha \mathbf{d}) < f(\mathbf{x}_0) \qquad \forall \alpha \in (0, \bar{\alpha}).$$

Lemma 3.1.18. *Siano $f \in C^1(\Omega)$ e $\mathbf{x} \in \Omega$. Ogni vettore $\mathbf{d} \in \mathbb{R}^n$ tale che*

$$\nabla f(\mathbf{x}) \cdot \mathbf{d} < 0$$

è una direzione di discesa di f in $\mathbf{x}$.

Vediamo prima la dimostrazione di questo lemma, che ci servirà per dimostrare il Teorema di Fermat.

Dimostrazione. Siano $x \in \Omega$, e g una funzione definita come $g(\alpha) = f(\mathbf{x}+\alpha\mathbf{d})$. Sia poi $\alpha_0 > 0$ tale che $\mathbf{x} + \alpha\mathbf{d} \in \Omega$ per ogni $\alpha \in (-\alpha_0, \alpha_0)$. Quindi $g : (-\alpha_0, \alpha_0) \to \mathbb{R}$. Applicando la regola della catena vediamo che $g \in C^1\Big((-\alpha_0, \alpha_0)\Big)$ con derivata data da

$$g'(\alpha) = \nabla f(\mathbf{x} + \alpha\mathbf{d}) \cdot \mathbf{d}.$$

Ora vogliamo capire se esiste un $\bar{\alpha} \in [0, \alpha_0)$ tale che $g(\alpha) < g(0)$ per ogni $\alpha \in [0, \bar{\alpha})$. In caso affermativo, ne seguirà che $\mathbf{d}$ è una direzione di discesa. Grazie alla formula dello sviluppo di Taylor al primo ordine vediamo che, per $\alpha \to 0$

$$\begin{aligned} & g(\alpha) = g(0) + g'(0)\alpha + o(\alpha) \\ \iff\ & f(\mathbf{x} + \alpha\mathbf{d}) = f(\mathbf{x}) + \alpha\nabla f(\mathbf{x}) \cdot \mathbf{d} + o(\alpha) \\ \iff\ & \frac{f(\mathbf{x} + \alpha\mathbf{d}) - f(\mathbf{x})}{\alpha} = \nabla f(\mathbf{x}) \cdot \mathbf{d} + \frac{o(\alpha)}{\alpha}. \end{aligned}$$

Poiché

$$\frac{o(\alpha)}{\alpha} = o(1),$$

possiamo rendere il termine di resto piccolo a piacere. Scegliamo $\bar{\alpha} > 0$ tale che:

$$\left|\frac{o(\alpha)}{\alpha}\right| < \frac{1}{2}|\nabla f(\mathbf{x}) \cdot \mathbf{d}| \qquad \forall\ \alpha \in (-\bar{\alpha}, \bar{\alpha}).$$

Questo serve per concludere che

$$\frac{f(\mathbf{x} + \alpha\mathbf{d}) - f(\mathbf{x})}{\alpha} < \nabla f(\mathbf{x}) \cdot \mathbf{d} + \frac{1}{2}|\nabla f(\mathbf{x}) \cdot \mathbf{d}|.$$

Quindi, se abbiamo $\nabla f(\mathbf{x}) \cdot \mathbf{d} < 0$, allora

$$\frac{f(\mathbf{x} + \alpha\mathbf{d}) - f(\mathbf{x})}{\alpha} < 0 \quad \implies \quad f(\mathbf{x} + \alpha\mathbf{d}) < f(\mathbf{x}) \qquad \forall\ \alpha \in (0, \bar{\alpha}),$$

cioè $\mathbf{d}$ è una direzione di discesa, come si voleva dimostrare. □

Grazie al Lemma 3.1.18 possiamo dimostrare il teorema di Fermat.

Dimostrazione. Assumiamo che $\mathbf{x}_0$ sia un minimo locale, se $\nabla f(\mathbf{x}_0) \neq 0$, allora esiste

$$i \in \{1, \dots, n\} \quad \text{tale che} \quad \frac{\partial f}{\partial x_i}(\mathbf{x}_0) \neq 0,$$

e osserviamo che

$$\nabla f(\mathbf{x}_0) \cdot \mathbf{e}_i = \frac{\partial f}{\partial x_i}(\mathbf{x}_0).$$

Ora, dato che questa derivata parziale è diversa da zero, abbiamo due casi:

1. Se $\frac{\partial f}{\partial x_i}(\mathbf{x}_0) > 0$, allora:
$$\nabla f(\mathbf{x}_0) \cdot (-\mathbf{e}_i) = -\frac{\partial f}{\partial x_i}(\mathbf{x}_0) < 0,$$
quindi $-\mathbf{e}_i$ è direzione di discesa.

2. Se $\frac{\partial f}{\partial x_i}(\mathbf{x}_0) < 0$, allora:
$$\nabla f(\mathbf{x}_0) \cdot \mathbf{e}_i = \frac{\partial f}{\partial x_i}(\mathbf{x}_0) < 0,$$
quindi $\mathbf{e}_i$ è direzione di discesa.

Quindi, $\mathbf{x}_0$ non può essere un minimo locale.
In caso di un massimo locale, lo si gestisce in modo analogo. Ricordando l'osservazione 3.1.2, possiamo dire che valgono le stesse considerazioni sul gradiente, sostituendo f con $-f$. □

Osservazione 3.1.19. La condizione del teorema di Fermat (anche noto come First Order Necessary condition) è necessaria, ma non sufficiente per garantire un minimo. Infatti, un punto critico può essere un minimo, un massimo o un punto di sella.

Ci serve quindi un modo per determinare se un punto critico sia effettivamente un minimo.

Condizioni di ottimalità del secondo ordine

Teorema 3.1.20 (Condizioni di ottimalità del secondo ordine). *Siano $f \in C^2(\Omega)$ e $\mathbf{x}_0 \in \Omega$. Supponiamo che $\mathbf{x}_0$ sia un punto critico, cioè*
$$\nabla f(\mathbf{x}_0) = 0.$$
Allora abbiamo i seguenti casi:

- *se la matrice Hessiana è definita positiva,*
$$\mathbf{H}f(\mathbf{x}_0) \succ 0,$$
allora $\mathbf{x}_0$ è un punto di minimo locale;
- *se la matrice Hessiana è definita negativa,*
$$\mathbf{H}f(\mathbf{x}_0) \prec 0,$$
allora $\mathbf{x}_0$ è un punto di massimo locale;
- *se la matrice Hessiana è non definita, allora $\mathbf{x}_0$ è un punto di sella.*

Osservazione 3.1.21. Se il gradiente è nullo e la matrice Hessiana è definita positiva o negativa, allora abbiamo una condizione sufficiente per l'ottimalità, e il punto è sicuramente un minimo o un massimo locale.
Se invece la matrice Hessiana è solo semidefinita, allora non si può concludere nulla, perché valgono solo le implicazioni opposte, ovvero: se il punto è di massimo locale, allora la matrice è semidefinita negative; se il punto è di minimo locale, allora la matrice è semidefinita negativa. Tuttavia, si può avere una matrice semidefinita positiva o negativa anche se il punto è di sella.

3.2 Convessità

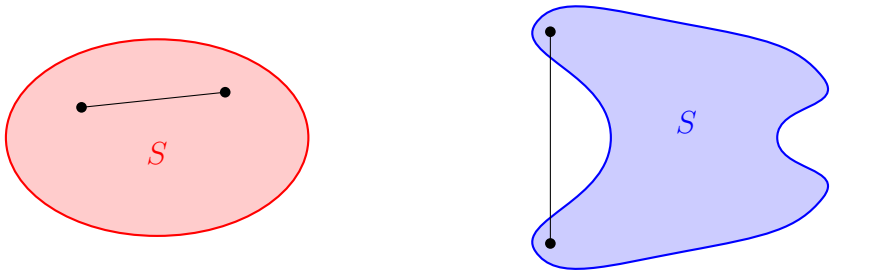
Le condizioni d'ottimalità viste finora permettono di individuare solo soluzioni ottime locali. In molti contesti applicativi abbiamo bisogno di soluzioni ottime globali. Per poter garantire che un punto di minimo locale sia anche un minimo globale, dobbiamo imporre ulteriori ipotesi sulla funzione obiettivo f e sul dominio ammissibile E. Tra queste, la convessità rappresenta la proprietà più potente e utile, in quanto consente di trasformare un problema locale in uno globale.

3.2.1 Fondamenti di convessità

Definizione 3.2.1 (**Insieme convesso**)**.** Un insieme $E \subseteq \mathbb{R}^n$ si dice convesso se, per ogni coppia di punti $\mathbf{x}, \mathbf{y} \in E$ e per ogni $\lambda \in [0,1]$, abbiamo

$$\lambda \mathbf{x} + (1-\lambda)\mathbf{y} \in E.$$

Geometricamente ciò significa che un insieme è convesso se il segmento che unisce due punti qualsiasi dell'insieme è completamente contenuto nell'insieme stesso.



Insieme convesso di $\mathbb{R}^2$ Insieme non convesso di $\mathbb{R}^2$

Figura 3.2: Esempio di insieme convesso e non convesso.

Definizione 3.2.2 (**Combinazione convessa**)**.** Data una collezione di vettori $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \mathbb{R}^n$, una combinazione convessa è una combinazione lineare

$$\lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2 + \cdots + \lambda_n \mathbf{x}_n,$$

dove

$$\sum_{i=1}^{n} \lambda_i = 1 \quad \text{e} \quad \lambda_i \geq 0 \quad \forall i \in \{1, \ldots, n\}.$$

In altre parole, una combinazione convessa è una combinazione lineare dei vettori $\mathbf{x}_i$ in cui i pesi sono non negativi e la loro somma è uguale a 1.

Osservazione 3.2.3. Siano $E \subseteq \mathbb{R}^n$ un insieme convesso, e $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in E$. Allora ogni combinazione convessa dei punti $\mathbf{x}_i$ appartiene all'insieme E.

Definizione 3.2.4 (**Funzione convessa**)**.** Sia $E \subseteq \mathbb{R}^n$ un insieme convesso. Una funzione $f : E \to \mathbb{R}$ si dice convessa se, per ogni $\mathbf{x}, \mathbf{y} \in E$ e per ogni $\lambda \in [0,1]$, abbiamo

$$f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}).$$

La disuguaglianza dice che il grafico della funzione si trova al di sotto del segmento che unisce i punti $\big(\mathbf{x}, f(\mathbf{x})\big) \in \mathbb{R}^{n+1}$ e $\big(\mathbf{y}, f(\mathbf{y})\big) \in \mathbb{R}^{n+1}$. In altre parole, il segmento di linea sta sopra il grafico della funzione.

Definizione 3.2.5 (**Funzione concava**)**.** Se f è convessa, allora $-f$ si dice concava.

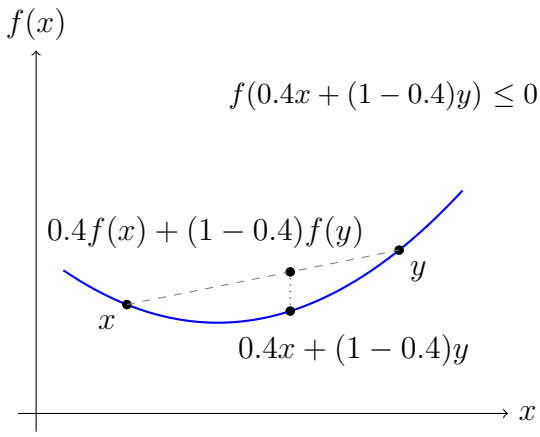


Figura 3.3: Rappresentazione grafica di una funzione convessa in $\mathbb{R}^2$

3.2.2 Convessità in una dimensione

Cominciamo con il caso monodimensionale, che è più semplice ma contiene già le idee fondamentali.

Teorema 3.2.6 (**Caratterizzazione della convessità in una dimensione**). *Siano $I \subseteq \mathbb{R}$ un intervallo aperto e $f \in C^2(I)$. Le seguenti affermazioni sono equivalenti:*

1. *f è convessa su I,*
2. *$f'(x)$ è crescente su I,*
3. *$f(y) \geq f(x) + f'(x)(y-x)$ per ogni $x, y \in I$,*
4. *$f''(x) \geq 0$ per ogni $x \in I$.*

Dimostrazione. **Equivalenza (2) $\Leftrightarrow$ (4):** Segue direttamente dalla relazione tra monotonia della derivata prima e segno della derivata seconda.

(1) $\Rightarrow$ (3): Siano $x, y \in I$ con $x \neq y$. Per definizione di convessità abbiamo che per ogni $\lambda \in (0,1)$ vale

$$f\Big(\lambda y + (1-\lambda)x\Big) \leq \lambda f(y) + (1-\lambda)f(x) = f(x) + \lambda\Big(f(y) - f(x)\Big).$$

Riscrivendo e dividendo per λ otteniamo

$$\frac{f\Big(x + \lambda(y-x)\Big) - f(x)}{\lambda} \leq f(y) - f(x).$$

Dividendo inoltre per $(y-x)$

$$\frac{f\Big(x + \lambda(y-x)\Big) - f(x)}{\lambda(y-x)} \leq \frac{f(y) - f(x)}{y-x},$$

e prendendo il limite per $\lambda \to 0^+$ il termine a sinistra tende a $f'(x)$ e otteniamo

$$f'(x)(y-x) \leq f(y) - f(x),$$

che equivale al punto (3).

(3) ⇒ (1): Siano $x, y \in I$ e $\lambda \in [0,1]$ e poniamo $z = \lambda x + (1-\lambda)y$, combinazione convessa di x e y.
Applicando la disuguaglianza del punto (3) in z otteniamo:

$$f(x) \geq f(z) + f'(z)(x-z),$$
$$f(y) \geq f(z) + f'(z)(y-z).$$

Moltiplicando la prima disuguaglianza per λ e la seconda per $(1-\lambda)$, e sommando, abbiamo

$$\lambda f(x) + (1-\lambda)f(y) \geq f(z) + f'(z)(\lambda x + (1-\lambda)y - z) = f(z),$$

che è proprio la definizione di convessità.

(2) ⇒ (3): Fissati due punti $x, y \in I$ con $x \neq y$, per il teorema di Lagrange, esiste ξ compreso tra x e y tale che:

$$f(y) - f(x) = f'(\xi)(y-x). \tag{3.2.1}$$

Distinguiamo allora tra due casi.

1. Se $y > x$, allora $\xi > x$ e, dato che f' è crescente, abbiamo $f'(\xi) \geq f'(x)$. Essendo $y - x > 0$, abbiamo
$$f(y) - f(x) = f'(\xi)(y-x) \geq f'(x)(y-x).$$
2. Se $y < x$, allora $\xi < x$ e $f'(\xi) \leq f'(x)$. Rimaneggiando la (3.2.1) otteniamo
$$f(x) - f(y) = f'(\xi)(x-y) \leq f'(x)(x-y),$$
e dato che $y - x < 0$
$$f(y) - f(x) \geq f'(x)(y-x).$$

In entrambi i casi:

$$f(y) - f(x) \geq f'(x)(y-x).$$

(3) ⇒ (2): Per $x < y$, grazie al punto (3) abbiamo:

$$f(y) \geq f(x) + f'(x)(y-x),$$
$$f(x) \geq f(y) + f'(y)(x-y).$$

Semplici manipolazioni algebriche permettono di riscrivere queste disuguaglianze come:

$$f(y) - f(x) \geq f'(x)(y-x),$$
$$f(y) - f(x) \leq f'(y)(y-x),$$

da cui segue $f'(x) \leq f'(y)$. □

Osservazione 3.2.7. La disuguaglianza del punto (3) del Teorema 3.2.10 ha un'importante interpretazione geometrica: il grafico di f si trova al di sopra di tutte le sue rette tangenti.

Concludiamo questa sezione con un lemma tecnico che permette di lavorare con funzioni convesse su intervalli chiusi.

Lemma 3.2.8. *Siano $a, b \in \mathbb{R}$, $a < b$, e sia $f \in C([a,b])$. Allora f è convessa su $[a,b]$ se e solo se è convessa su (a,b).*

Dimostrazione. Chiaramente, dobbiamo solo mostrare che, se f è convessa su (a,b), allora è convessa anche sull'intervallo chiuso $[a,b]$. Dato che è convessa su (a,b), sappiamo che

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$$

per ogni $x, y \in (a,b)$ e ogni $\lambda \in [0,1]$. Allora, per la continuità su $[a,b]$, possiamo prendere il limite per $x \to a^+$ oppure per $y \to b^-$, oppure entrambi contemporaneamente, e così possiamo concludere la disuguaglianza di convessità in realtà vale su tutto $[a,b]$. □

3.2.3 Funzioni convesse in più variabili

Una proprietà fondamentale della convessità, che utilizzeremo in seguito, è la seguente equivalenza.

Teorema 3.2.9 (**Convessità lungo le linee**). *Sia $E \subseteq \mathbb{R}^n$ un insieme convesso. Una funzione $f : E \to \mathbb{R}$ è convessa se e solo se per ogni $\mathbf{x}, \mathbf{y} \in E$, la funzione $\phi : [0,1] \to \mathbb{R}$ definita da*

$$\phi(t) = f\Big(\mathbf{x} + t(\mathbf{y} - \mathbf{x})\Big) = f\Big((1-t)\mathbf{x} + t\mathbf{y}\Big)$$

è convessa in $[0,1]$.

In termini intuitivi, questa proprietà afferma che una funzione $f : E \to \mathbb{R}$ è convessa se, fissato un qualsiasi punto $\mathbf{x}$ e una qualsiasi direzione $\mathbf{d} \in \mathbb{R}^n$, la funzione ottenuta "tagliando" f lungo la retta $t \mapsto \mathbf{x} + t\mathbf{d}$ è convessa come funzione reale.

Dimostrazione. Se f è convessa, allora per ogni $t_1, t_2 \in [0,1]$ e $\lambda \in [0,1]$ vale

$$\phi\Big(\lambda t_1 + (1-\lambda)t_2\Big) = f\Big((1 - [\lambda t_1 + (1-\lambda)t_2])\mathbf{x} + [\lambda t_1 + (1-\lambda)t_2]\mathbf{y}\Big),$$

e quindi, riordinando i termini in modo ottenere combinazioni convesse di $\mathbf{x}$ e $\mathbf{y}$ con parametri t_1, t_2, otteniamo

$$\phi(\lambda t_1 + (1-\lambda)t_2) = f(\lambda[(1-t_1)\mathbf{x} + t_1\mathbf{y}] + (1-\lambda)[(1-t_2)\mathbf{x} + t_2\mathbf{y}]).$$

Applichiamo la convessità di f per ottenere

$$\phi\Big(\lambda t_1 + (1-\lambda)t_2\Big) \leq \lambda f\Big([(1-t_1)\mathbf{x} + t_1\mathbf{y}]\Big) + (1-\lambda) f\Big([(1-t_2)\mathbf{x} + t_2\mathbf{y}]\Big),$$

che per la definizione di ϕ è

$$\phi\Big(\lambda t_1 + (1-\lambda)t_2\Big) \leq \lambda\phi(t_1) + (1-\lambda)\phi(t_2).$$

Quindi ϕ è convessa su $[0,1]$.

Supponiamo ora invece che per ogni $\mathbf{x}, \mathbf{y} \in E$ la funzione

$$\phi(t) = f\Big((1-t)\mathbf{x} + t\mathbf{y}\Big) \quad t \in [0,1],$$

sia convessa.
Allora per ogni $\lambda \in [0,1]$, per la convessità di ϕ applicata ai punti 0 e 1, otteniamo

$$\phi(\lambda) \leq (1-\lambda)\phi(0) + \lambda\phi(1),$$

da cui, applicando la definizione di ϕ, segue che

$$f\Big((1-\lambda)\mathbf{x} + \lambda\mathbf{y}\Big) \leq (1-\lambda) f(\mathbf{x}) + \lambda f(\mathbf{y}),$$

che è la definizione di convessità di f su E. □

3.2.4 Estensione al caso multidimensionale

Teorema 3.2.10 (**Caratterizzazioni della convessità in più variabili**). *Siano $E \subseteq \mathbb{R}^n$ un insieme aperto e convesso, e $f \in C^2(E)$. Allora le seguenti affermazioni sono equivalenti:*

1. *f è convessa su E,*
2. *$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x})$ per ogni $\mathbf{x}, \mathbf{y} \in E$,*
3. *$\mathbf{H}f(\mathbf{x}) \succeq 0$ per ogni $\mathbf{x} \in E$.*

Dimostrazione. **Equivalenza (1) $\Leftrightarrow$ (2):** La dimostrazione è analoga al caso monodimensionale.

(1) $\Rightarrow$ (2): Per $\mathbf{x}, \mathbf{y} \in E$ con $\mathbf{x} \neq \mathbf{y}$ e $\lambda \in (0, 1]$, per la definizione di convessità otteniamo

$$f\Big(\lambda\mathbf{y} + (1-\lambda)\mathbf{x}\Big) \leq \lambda f(\mathbf{y}) + (1-\lambda) f(\mathbf{x}) = f(\mathbf{x}) + \lambda\Big(f(\mathbf{y} - f(\mathbf{x})\Big).$$

Riscrivendo e dividendo per λ vediamo che

$$\frac{f\Big(\mathbf{x} + \lambda(\mathbf{y} - \mathbf{x})\Big) - f(\mathbf{x})}{\lambda} \leq f(\mathbf{y}) - f(\mathbf{x}).$$

Prendendo il limite per $\lambda \to 0^+$, notiamo che il termine a sinistra è la definizione di derivata direzionale

$$\partial_{(\mathbf{y}-\mathbf{x})} f(\mathbf{x}) \leq f(\mathbf{y}) - f(\mathbf{x}).$$

Applicando la relazione tra derivata direzionale e gradiente (Teorema 2.3.11), risulta

$$\nabla f(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x}) \leq f(\mathbf{y}) - f(\mathbf{x}).$$

(2) $\Rightarrow$ (1): Siano $\mathbf{x}, \mathbf{y} \in E$ e $\lambda \in [0, 1]$. Poniamo $\mathbf{z} = \lambda\mathbf{x} + (1-\lambda)\mathbf{y}$. Applicando la disequazione del punto (2) in $\mathbf{z}$ abbiamo

$$f(\mathbf{x}) \geq f(\mathbf{z}) + \nabla f(\mathbf{z}) \cdot (\mathbf{x} - \mathbf{z}),$$
$$f(\mathbf{y}) \geq f(\mathbf{z}) + \nabla f(\mathbf{z}) \cdot (\mathbf{y} - \mathbf{z}).$$

Moltiplicando la prima disuguaglianza per λ, la seconda per $(1-\lambda)$ e poi sommando le due, otteniamo

$$\lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y}) \geq f(\mathbf{z}) + \nabla f(\mathbf{z}) \cdot (\lambda\mathbf{x} + (1-\lambda)\mathbf{y} - \mathbf{z}) = f(\mathbf{z}).$$

Equivalenza (1) $\Rightarrow$ (3): Per il teorema sulla convessità lungo le linee, f è convessa se e solo se per ogni $\mathbf{x}, \mathbf{y} \in E$ la funzione

$$\phi(t) = f\Big(\mathbf{x} + t(\mathbf{y} - \mathbf{x})\Big),$$

è convessa in una variabile.
Poiché $f \in C^2(E)$, allora $\phi \in C^2((0,1))$ e, per il Teorema 3.2.6 (convessità $\Leftrightarrow$ derivata seconda non negativa), ϕ è convessa se e solo se $\phi''(t) \geq 0$ per ogni $t \in (0,1)$. La regola della catena (Proposizione 2.5.4) fornisce

$$\phi'(t) = \nabla f\Big(\mathbf{x} + t(\mathbf{y} - \mathbf{x})\Big) \cdot (\mathbf{y} - \mathbf{x}),$$
$$\phi''(t) = (\mathbf{y} - \mathbf{x})^\top \mathbf{H}f\Big(\mathbf{x} + t(\mathbf{y} - \mathbf{x})\Big)(\mathbf{y} - \mathbf{x}).$$

In particolare, prendendo il limite per $t \to 0^+$ e notando che $\mathbf{y} - \mathbf{x}$ è arbitrario, otteniamo $\mathbf{v}^\top \mathbf{H}f(\mathbf{x})\mathbf{v} \geq 0$ per ogni $\mathbf{v} \in \mathbb{R}^n$, e dunque $\mathbf{H}f(x)$ è semidefinita positiva.

Equivalenza (3) $\Rightarrow$ (1): Dati $\mathbf{x}, \mathbf{y} \in E$, consideriamo la funzione

$$\phi(t) = f\Big((1-t)\mathbf{x} + t\mathbf{y}\Big).$$

Applicando la regola della catena come nel punto precedente, vediamo che

$$\phi''(t) = (\mathbf{y} - \mathbf{x})^\top \mathbf{H}f\Big(\mathbf{x} + t(\mathbf{y} - \mathbf{x})\Big)(\mathbf{y} - \mathbf{x}).$$

Per ipotesi $\mathbf{H}f(z) \succeq 0$ per ogni $z \in E$, dunque $\phi''(t) \geq 0$ per ogni $t \in (0,1)$. Quindi, per la caratterizzazione monodimensionale della convessità (Teorema 3.2.6), e per il Lemma 3.2.8, ϕ è convessa su $[0,1]$, e dunque ne segue che f è convessa, grazie al Teorema 3.2.9. □

Osservazione 3.2.11. La disuguaglianza del punto (2) del teorema 3.2.10 significa che il piano tangente al grafico di f in qualsiasi punto $\mathbf{x}$ sta sempre al di sotto del grafico della funzione.

3.2.5 Implicazioni per l'ottimizzazione

I risultati sulla convessità hanno importanti conseguenze per i problemi di ottimizzazione.

Teorema 3.2.12 (**Proprietà fondamentale della convessità**)**.** *Consideriamo un problema di ottimizzazione:*

$$\min_{\mathbf{x} \in E} f(\mathbf{x}),$$

dove $E \subseteq \mathbb{R}^n$ è un insieme convesso e $f : E \to \mathbb{R}$ è una funzione convessa. Allora ogni minimo locale $\mathbf{x}_0$ di f in E è anche un minimo globale, ovvero

$$f(\mathbf{x}_0) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \in E.$$

Dimostrazione. Sia $\mathbf{x}_0 \in E$ un minimo locale, allora per definizione esiste un $\varepsilon > 0$ tale che

$$f(\mathbf{x}_0) \leq f(\mathbf{x}) \quad \forall\ \mathbf{x} \in E \cap B(\mathbf{x}_0, \varepsilon).$$

Sia ora $\mathbf{y} \in E$. Poiché f è convessa, per ogni $\lambda \in [0,1]$ abbiamo

$$f(\lambda \mathbf{y} + (1-\lambda)\mathbf{x}_0) \leq \lambda f(\mathbf{y}) + (1-\lambda) f(\mathbf{x}_0).$$

Definiamo la combinazione convessa

$$\mathbf{x}_\lambda = \lambda \mathbf{y} + (1-\lambda)\mathbf{x}_0,$$

che per definizione appartiene a E. Inoltre, $\mathbf{x}_\lambda \to \mathbf{x}_0$ quando $\lambda \to 0^+$, quindi per λ sufficientemente piccolo, $\mathbf{x}_\lambda \in B(\mathbf{x}_0, \varepsilon)$.
Quindi, se λ è sufficientemente piccolo, per definizione di minimo locale

$$f(\mathbf{x}_0) \leq f(\mathbf{x}_\lambda).$$

Quindi per tali valori di λ abbiamo

$$f(\mathbf{x}_0) \leq f(\mathbf{x}_\lambda) \leq \lambda f(\mathbf{y}) + (1-\lambda) f(\mathbf{x}_0).$$

Sottraiamo $(1-\lambda) f(\mathbf{x}_0)$ da entrambi i membri e otteniamo

$$\lambda f(\mathbf{x}_0) \leq \lambda f(\mathbf{y})$$

A questo punto, è sufficiente dividere per λ (che è positivo), per vedere che

$$f(\mathbf{x}_0) \leq f(\mathbf{y}),$$

e quindi, dato che $\mathbf{y} \in E$ è un punto arbitrario, concludiamo che $\mathbf{x}_0$ è un minimo globale.
□

Oltre alla semplice convessità, esistono proprietà più restrittive che facilitano ulteriormente l'individuazione del minimo globale di una funzione.

3.2.6 Convessità rafforzata

Illustriamo ora due nozioni che rafforzano il concetto di convessità.

Definizione 3.2.13 (**Funzione strettamente convessa**)**.** Sia $E \subseteq \mathbb{R}^n$ un insieme convesso. Una funzione $f : E \to \mathbb{R}$ è detta strettamente convessa se, per ogni $\mathbf{x}, \mathbf{y} \in E$ con $\mathbf{x} \neq \mathbf{y}$ e per ogni $\lambda \in (0, 1)$, abbiamo

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) < \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y}).$$

In parole povere, la differenza tra convessità e stretta convessità è semplicemente che la disuguaglianza nella definizione è stretta nel secondo caso (tranne ovviamente nei casi limite $\lambda = 0$ e $\lambda = 1$). Rimane sempre valida l'intuizione geometrica di prima; stavolta il segmento che unisce due punti del grafico della funzione sta strettamente sopra il grafico stesso (naturalmente tranne che nei punti estremi).

Teorema 3.2.14 (**Unicità del minimo per funzioni strettamente convesse**)**.** *Sia $E \subseteq \mathbb{R}^n$ un insieme convesso e sia $f : E \to \mathbb{R}$ una funzione strettamente convessa. Se f ammette un punto di minimo globale su E, allora questo punto è unico.*

Dimostrazione. Supponiamo, per assurdo, che esistano due punti distinti $\mathbf{x}_0 \in E$ e $\mathbf{x}_1 \in E$, con $\mathbf{x}_0 \neq \mathbf{x}_1$, tali che

$$f(\mathbf{x}_0) = f(\mathbf{x}_1) = \min_{\mathbf{x} \in E} f(\mathbf{x}) = a.$$

Consideriamo il punto interno al segmento $\mathbf{x}_\lambda = \lambda\mathbf{x}_0 + (1 - \lambda)\mathbf{x}_1$, per un lambda qualsiasi $\lambda \in (0, 1)$. Poiché E è convesso, $\mathbf{x}_\lambda \in E$. E per la stretta convessità di f, abbiamo:

$$f(\mathbf{x}_\lambda) < \lambda f(\mathbf{x}_0) + (1 - \lambda)f(\mathbf{x}_1) = \lambda a + (1 - \lambda)a = a.$$

Ma questo implica $f(\mathbf{x}_\lambda) < a$, il ché contraddice l'ipotesi che a sia il valore minimo globale su E. □

Vediamo ora la seconda nozione di convessità rafforzata.

Definizione 3.2.15 (**Funzione fortemente convessa**)**.** Sia $E \subseteq \mathbb{R}^n$ un insieme convesso. Una funzione $f : E \to \mathbb{R}$ è fortemente convessa se esiste un parametro $m > 0$ tale che:

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y}) - \frac{m}{2}\lambda(1 - \lambda)||\mathbf{x} - \mathbf{y}||^2 \quad \forall\, \mathbf{x}, \mathbf{y} \in \Omega \quad \text{e} \quad \lambda \in [0, 1].$$

Osservazione 3.2.16. Mentre il grafico di una funzione convessa normale sta sempre sotto il segmento che congiunge due punti, per una funzione fortemente convessa, tra il grafico e qualsiasi corda c'è sempre almeno uno spazio a forma di parabola.

Lemma 3.2.17. *Siano $E \subseteq \mathbb{R}^n$ un insieme convesso, e $f : E \to \mathbb{R}$. Allora valgono le seguenti implicazioni:*

- *se f è fortemente convessa, allora f è anche strettamente convessa;*
- *se f è strettamente convessa, allora f è anche convessa.*

In simboli,

$$\textit{Forte convessità} \implies \textit{Stretta convessità} \implies \textit{Convessità}.$$

Dimostrazione. Dimostriamo per prima cosa che la forte convessità implica la stretta convessità. Se f è fortemente convessa allora, per ogni $\mathbf{x}, \mathbf{y} \in E$, $\mathbf{x} \neq \mathbf{y}$ e $\lambda \in (0,1)$, abbiamo

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}) - \frac{m}{2}\lambda(1-\lambda)||\mathbf{x}-\mathbf{y}||^2.$$

Poiché $m > 0$ e $||\mathbf{x}-\mathbf{y}||^2 > 0$, il termine sottratto è positivo e otteniamo la disuguaglianza stretta

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) < \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}).$$

Quindi f è strettamente convessa.
Ora mostriamo che la stretta convessità implica la convessità. La definizione di convessità richiede la disuguaglianza non stretta

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}).$$

Se la disuguaglianza stretta è soddisfatta per ogni $\mathbf{x} \neq \mathbf{y}$, allora anche la versione non stretta è soddisfatta, mentre nei casi $\mathbf{x} = \mathbf{y}$, $\lambda = 0$ e $\lambda = 1$ è banalmente vera. □

Teorema 3.2.18. *Siano $\Omega \subseteq \mathbb{R}^n$ un insieme convesso e sia $f \in C^1(\Omega)$. Allora f è fortemente convessa con costante $m > 0$ se e solo se*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot (\mathbf{y}-\mathbf{x}) + \frac{m}{2}||\mathbf{y}-\mathbf{x}||^2 \quad \textit{per ogni} \quad \mathbf{x}, \mathbf{y} \in \Omega.$$

Concludiamo il capitolo fornendo una caratterizzazione della forte convessità per funzioni C^2 in termini del comportamento della matrice Hessiana.

Teorema 3.2.19 (**Condizione del secondo ordine equivalente**)**.** *Siano $\Omega \subseteq \mathbb{R}^n$ un insieme aperto convesso e $f \in C^2(\Omega)$. Allora f è fortemente convessa con costante $m > 0$ se e solo se la sua matrice Hessiana soddisfa*

$$\mathbf{H}f(\mathbf{x}) \succeq m\mathbf{I} \qquad \forall\ \mathbf{x} \in \Omega,$$

cioè $\mathbf{H}f(\mathbf{x}) - m\mathbf{I}$ è semidefinita positiva per ogni $\mathbf{x} \in \Omega$.

Dimostrazione. **(⇒)** Supponiamo $\mathbf{H}f(\mathbf{z}) \succeq m\mathbf{I}$ per ogni $\mathbf{z} \in \Omega$. Siano $\mathbf{x}, \mathbf{y} \in \Omega$ e $\mathbf{d} = \mathbf{y} - \mathbf{x}$. Definiamo $\varphi(t) = f(\mathbf{x} + t\mathbf{d})$ per $t \in [0,1]$. Per il teorema dello sviluppo di Taylor in una variabile con resto di Lagrange, esiste $\xi \in (0,1)$ tale che

$$\varphi(1) = \varphi(0) + \varphi'(0) + \tfrac{1}{2}\varphi''(\xi).$$

Ora

$$\varphi'(0) = \nabla f(\mathbf{x}) \cdot \mathbf{d}$$

e

$$\varphi''(\xi) = \mathbf{d}^\top \mathbf{H}f(\mathbf{x} + \xi\mathbf{d})\mathbf{d}.$$

Per ipotesi

$$\varphi''(\xi) \geq m\|\mathbf{d}\|^2,$$

quindi, sostituendo ϕ e ϕ' nello sviluppo di Taylor, otteniamo

$$f(\mathbf{y}) = f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot (\mathbf{y}-\mathbf{x}) + \tfrac{1}{2}\varphi''(\xi) \geq f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot (\mathbf{y}-\mathbf{x}) + \tfrac{m}{2}\|\mathbf{y}-\mathbf{x}\|^2,$$

cioè f è fortemente convessa con costante m.

($\Leftarrow$) Supponiamo ora che f sia fortemente convessa con costante $m > 0$, ossia

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x}) \cdot (\mathbf{y} - \mathbf{x}) + \frac{m}{2}\|\mathbf{y} - \mathbf{x}\|^2 \qquad \forall\, \mathbf{x}, \mathbf{y} \in \Omega.$$

Fissiamo $\mathbf{x} \in \Omega$ e una direzione arbitraria $\mathbf{v} \in \mathbb{R}^n$. Per t sufficientemente piccolo, $\mathbf{x} + t\mathbf{v} \in \Omega$, e quindi

$$f(\mathbf{x} + t\mathbf{v}) \geq f(\mathbf{x}) + t\,\nabla f(\mathbf{x}) \cdot \mathbf{v} + \frac{m}{2}t^2\|\mathbf{v}\|^2.$$

Dividendo per t^2 (con $t \neq 0$), si ottiene

$$\frac{f(\mathbf{x} + t\mathbf{v}) - f(\mathbf{x}) - t\,\nabla f(\mathbf{x}) \cdot \mathbf{v}}{t^2} \geq \frac{m}{2}\|\mathbf{v}\|^2.$$

Poiché $f \in C^2(\Omega)$, passando al limite per $t \to 0$,

$$\frac{1}{2}\mathbf{v}^\top \mathbf{H} f(\mathbf{x})\, \mathbf{v} \geq \frac{m}{2}\|\mathbf{v}\|^2.$$

Moltiplicando entrambi i membri per 2,

$$\mathbf{v}^\top \mathbf{H} f(\mathbf{x})\, \mathbf{v} \geq m\|\mathbf{v}\|^2 \qquad \forall\, \mathbf{v} \in \mathbb{R}^n.$$

Da questa disuguaglianza, che vale per ogni $\mathbf{v}$, segue che

$$\mathbf{H} f(\mathbf{x}) \succeq m\mathbf{I} \qquad \forall\, \mathbf{x} \in \Omega.$$

□

Osservazione 3.2.20. L'ordinamento $\succeq$ indica che $\mathbf{H} f(\mathbf{x}) - m\mathbf{I}$ è semidefinita positiva. Il che è equivalente a dire che la forma quadratica di $\mathbf{H} f(\mathbf{x}) - m\mathbf{I}$ è positiva, cioè

$$\mathbf{v}^\top \mathbf{H} f(\mathbf{x}) \mathbf{v} \geq m\|\mathbf{v}\|^2 \quad \forall\ \mathbf{v} \in \mathbb{R}^n.$$

Capitolo 4

Reti Neurali Artificiali

In questo capitolo analizziamo la struttura matematica e il funzionamento delle reti neurali artificiali, modelli che costituiscono il cuore dell'apprendimento automatico. Partendo dal singolo neurone fino alle architetture multilivello, ne studieremo il comportamento formale, le funzioni di attivazione e il legame con i teoremi di approssimazione.

4.1 Struttura e funzionamento

Nella programmazione convenzionale, il programmatore fornisce al computer istruzioni esplicite, suddividendo un problema complesso in una sequenza di compiti elementari ben definiti. Le reti neurali artificiali, invece, non richiedono che l'algoritmo venga programmato per risolvere direttamente il problema: esse imparano a farlo osservando i dati. In questo modo è il modello stesso a determinare una soluzione, adattando i propri parametri in base a quanto ha imparato.

Ormai esistono tante varianti di reti neurali, come le reti neurali convoluzionali e le reti neurali ricorrenti, giusto per dirne un paio.
Per capire il funzionamento principale però, è meglio guardare la rete neurale più semplice, i cosiddetti percettroni multistrati.
Prima di esaminare la rete neurale nel suo complesso, dobbiamo comprenderne gli elementi costitutivi, chiamati neuroni o percettroni.

4.2 Neuroni artificiali

Il termine "rete neurale" nasce dal tentativo di imitare, almeno in parte, il funzionamento del cervello umano. Naturalmente, i modelli utilizzati in informatica non hanno la complessità né la fedeltà biologica del sistema nervoso, ma riprendono l'idea fondamentale: un insieme di unità semplici (i neuroni artificiali) che, collegate tra loro, sono in grado di risolvere compiti complessi.

In termini matematici, un neurone artificiale è un modello computazionale che elabora un certo numero di valori in ingresso, ciascuno pesato in base alla sua importanza, e produce un singolo valore in uscita tramite una trasformazione non lineare. Questo meccanismo di base, seppur semplice, diventa estremamente potente quando milioni di tali unità sono interconnesse in una rete.

4.2.1 Percettrone

Il percettrone è il "mattone fondamentale" di una rete neurale. Riceve un certo numero n di valori in ingresso $x_1, x_2, \ldots, x_n \in \mathbb{R}$ e produce un'unica uscita. Ogni ingresso è associato a un

peso $\omega_1, \omega_2, \ldots, \omega_n \in \mathbb{R}$, che ne controlla l'importanza: durante l'apprendimento, questi pesi vengono modificati per migliorare le prestazioni della rete.

La prima operazione svolta dal percettrone è calcolare la somma pesata degli ingressi:

$$\sum_{i=1}^{n} \omega_i x_i.$$

Inoltre, ogni percettrone dispone di un parametro additivo detto bias $b \in \mathbb{R}$, che consente di traslare la funzione di attivazione e di aumentare la flessibilità del modello. Senza il bias, infatti, la funzione appresa sarebbe vincolata a passare per l'origine.

La somma pesata degli ingressi, arricchita dal bias, è:

$$z = \sum_{i=1}^{n} \omega_i x_i + b.$$

Il risultato z viene poi trasformato da una funzione di attivazione $f(\cdot)$, che è un elemento fondamentale del percettrone. Essa introduce una componente non lineare che permette al modello di rappresentare relazioni più complesse tra input e output, superando i limiti delle semplici combinazioni lineari. Approfondiremo in dettaglio il ruolo e le principali tipologie di funzioni di attivazione nella sezione 4.7. L'uscita finale del percettrone è dunque:

$$a = f(z).$$

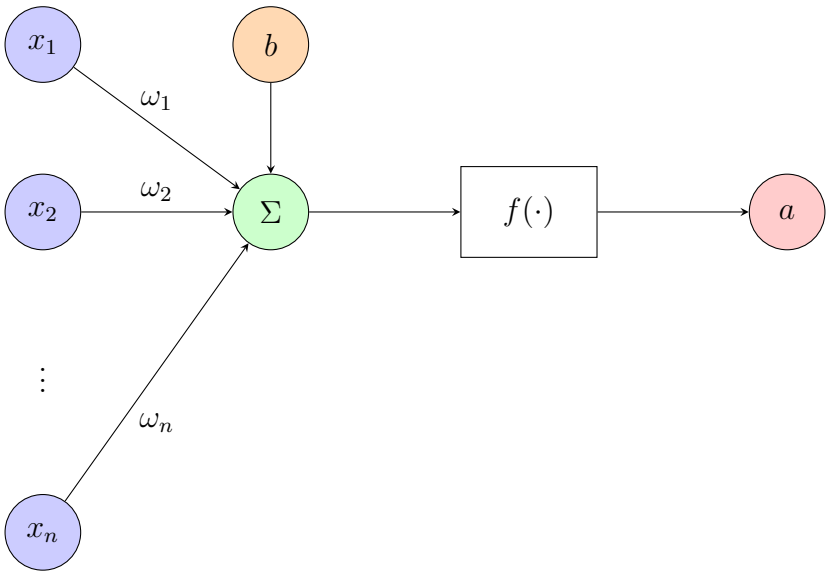


Figura 4.1: Schema di un percettrone con n ingressi, pesi ω_i e bias b, con colori distintivi per ogni componente.

4.3 Dal percettrone alla rete neurale

Un singolo percettrone, per quanto utile, ha delle capacità molto limitate. Per risolvere problemi più complessi, è necessario collegare molti percettroni in layer successivi, ottenendo così una rete neurale artificiale a più livelli, detta Multi Layer Perceptron (MLP).

Una MLP è una rete neurale feedforward, in cui i segnali si propagano dall'input verso l'output senza cicli o connessioni ricorrenti. Essa è tipicamente suddivisa in tre tipi di livelli principali.

4.3.1 Livello di input (Input layer)

Il livello di input di una MLP riceve i dati grezzi in ingresso e li trasmette ai neuroni del primo livello nascosto. I neuroni di questo livello non eseguono alcun calcolo, ma si limitano a trasmettere i valori di input ai neuroni nel primo livello nascosto.

4.3.2 Hidden layers

I layer nascosti contengono neuroni che combinano linearmente gli input ricevuti dal layer precedente e applicano una funzione di attivazione non lineare. L'introduzione di queste non-linearità consente alla rete di modellare relazioni complesse e di approssimare un'ampia classe di funzioni.

4.3.3 Output layer

Il livello di output di una MLP produce le previsioni finali o gli output della rete. Il numero di neuroni nel livello di output dipende dall'attività svolta (ad esempio, classificazione binaria, classificazione multipla, regressione). Ogni neurone nel livello di output riceve input dai neuroni nell'ultimo livello nascosto e applica una funzione di attivazione. Questa funzione di attivazione è solitamente diversa da quelle utilizzate nei livelli nascosti e produce il valore di output finale o la previsione.

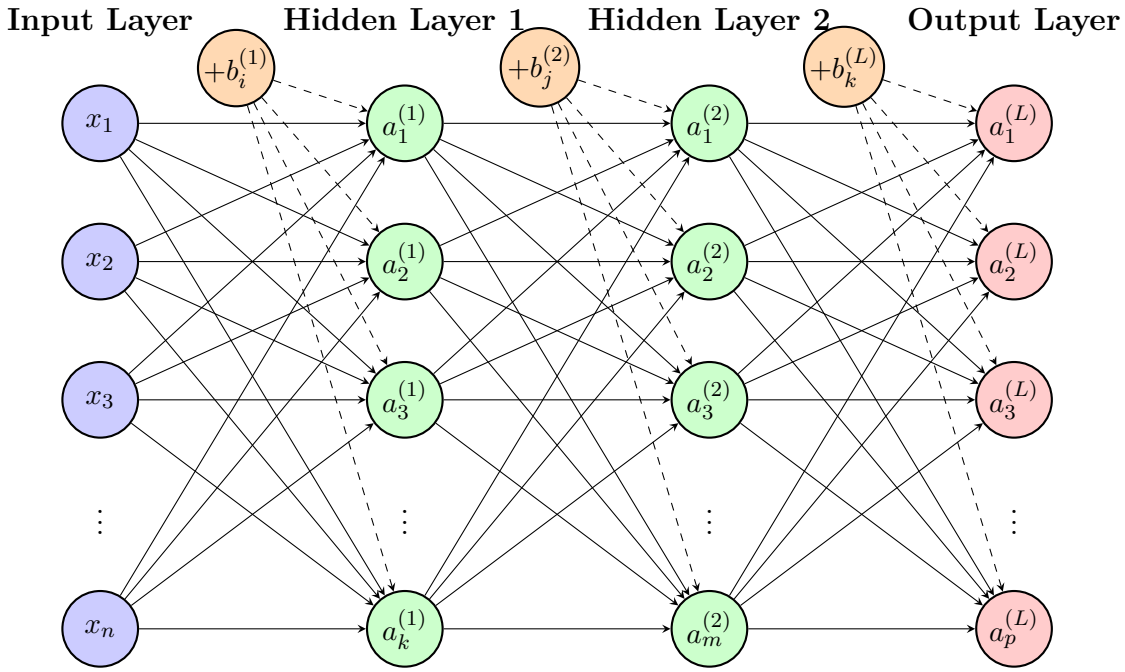


Figura 4.2: Schema di un Multi Layer Perceptron (MLP) con due layer nascosti e nodi di bias (connessioni tratteggiate).

Ci serve un modo per descrivere individualmente ogni singolo neurone di ogni stato. Perciò in questa tesi useremmo la seguente notazione.

Notazione 4.3.1. In questa tesi per i vari

- pesi $\omega_{ki}^{(l)}$ (collega neurone i del layer $l-1$ al neurone k del layer l,
- bias $b_i^{(l)}$,

- input $z_i^{(l)}$,
- output $a_i^{(l)}$,

dei vari layer:

- l'apice $l = 1, \ldots, L$, dove L è l'output layer, indica il layer (L è l'ultimo, cioè l'output layer),
- il pedice indica il neurone di tale layer.

Nei capitoli successivi approfondiremo come, tramite la backpropagation e la discesa del gradiente, la rete possa apprendere autonomamente i propri parametri per minimizzare la funzione obiettivo.

4.4 Teorema dell'Approssimazione Universale

Il Teorema dell'Approssimazione Universale (Universal Approximation Theorem) stabilisce che una rete neurale feedforward con un singolo layer nascosto, dotata di un numero sufficiente di neuroni e di una funzione di attivazione non lineare appropriata, è in grado di approssimare qualsiasi funzione continua con precisione arbitraria, purché la funzione sia definita su un insieme compatto di $\mathbb{R}^n$.

Questo risultato, dimostrato in forma rigorosa da Cybenko nel 1989 utilizzando funzioni di attivazione di tipo sigmoidale, e il l'ipercubo unitario n-dimensionale come dominio, rappresenta il fondamento teorico che giustifica l'utilizzo delle reti neurali come approssimatori di funzioni universali.

Teorema 4.4.1 (**Teorema dell'Approssimazione Universale (Cybenko, 1989)**). *Sia* $\sigma : \mathbb{R} \to \mathbb{R}$ *una funzione sigmoidale continua e limitata, data da*

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Sia $I_n = [0, 1]^n$ *l'ipercubo unitario n-dimensionale, cioè l'insieme di tutti i punti in* $\mathbb{R}^n$ *le cui coordinate sono comprese tra* 0 *e* 1

$$I_n = [0, 1]^n = \{(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n | 0 \leq x_i \leq 1 \text{ per ogni } i = 1, \ldots, n\}.$$

Allora, per ogni $f \in C(I_n)$ *e per ogni* $\epsilon > 0$*, esistono un intero* N*, pesi* $\omega_j \in \mathbb{R}^n$*, bias* $b_j \in \mathbb{R}$*, e coefficienti* $c_j \in \mathbb{R}$ *(in questo caso rappresentano i pesi del layer di output) tali che la funzione* $F : I_n \to \mathbb{R}$ *definita da*

$$F(\mathbf{x}) = \sum_{j=1}^{N} c_j \, \sigma(\omega_j \cdot \mathbf{x} + b_j)$$

soddisfa

$$\sup_{\mathbf{x} \in I_n} |F(\mathbf{x}) - f(\mathbf{x})| < \epsilon.$$

Il teorema garantisce che, almeno in linea di principio, una rete neurale con architettura sufficientemente ampia può rappresentare qualsiasi relazione funzionale continua tra input e output.

4.4.1 Limitazioni pratiche

Sebbene il teorema fornisca una garanzia teorica fondamentale, esso presenta diverse limitazioni pratiche di cruciale importanza.

Il teorema garantisce l'esistenza di una rete in grado di approssimare la funzione target, ma non fornisce un algoritmo per determinarne i parametri in modo efficiente. L'addestramento mediante discesa del gradiente potrebbe non convergere alla soluzione ottimale.

Inoltre, anche se teoricamente è sufficiente un unico hidden layer, il numero di neuroni N necessario per raggiungere una precisione ϵ potrebbe essere molto elevato, rendendo la rete computazionalmente inefficiente e soggetta a overfitting.

Un'ulteriore limitazione riguarda la generalizzazione: anche se il nostro modello approssima alla perfezione il training set, non è detto che tale approssimazione sia generalizzabile. Il teorema riguarda esclusivamente l'approssimazione sull'insieme compatto di definizione, tipicamente identificato con i dati di training, e non fornisce alcuna garanzia sulle prestazioni della rete su dati non visti.

Consideriamo infine la limitazione sull'addestrabilità: anche quando una soluzione esiste, il processo di ottimizzazione potrebbe essere ostacolato da problemi di ottimizzazione non convessi, gradienti instabili o inizializzazioni sfavorevoli.

Nonostante queste limitazioni, il teorema dell'approssimazione universale rimane un risultato fondamentale che giustifica l'utilizzo delle reti neurali per l'approssimazione di funzioni complesse in una vasta gamma di applicazioni pratiche.

4.5 Funzionamento di un Multi Layer Perceptron

Un Multi Layer Perceptron (MLP) è una rete neurale feedforward: i dati si propagano in avanti dal livello di input a quello di output attraversando uno o più layer nascosti.

4.5.1 Forward pass

Il forward pass è il processo mediante il quale i dati di input vengono propagati attraverso la rete neurale, layer dopo layer, fino a produrre l'output finale. Durante questa fase, la rete non apprende ancora: si limita ad applicare le trasformazioni matematiche determinate dai pesi e dai bias che definiscono il suo stato attuale. Il forward pass è quindi fondamentale per calcolare la previsione della rete.

I dati in ingresso partono dallo input layer, attraversano uno o più hidden layers e arrivano infine all'output layer.

4.6 Formalizzazione matematica del forward pass

Sia $\mathbf{x} \in \mathbb{R}^{n_0}$ il vettore di input alla rete, dove n_0 è il numero di neuroni nel input layer. E consideriamo una rete con L layer, dove il numero di neuroni per ogni layer può essere variabile, incluso quello di output. Per ogni layer $l \in \{1, 2, \ldots, L\}$ indichiamo con:

- $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ la matrice dei pesi, dove ogni riga contiene i pesi che collegano tutti i neuroni dello layer precedente ai neuroni dello layer corrente;
- $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$ il vettore dei bias;
- $f^{(l)}(\cdot)$ la funzione di attivazione applicata al layer l.

Il calcolo in avanti procede nel seguente modo. Per ogni neurone k-esimo dello layer l, si calcola il valore di ingresso (combinazione lineare degli output dei neuroni precedenti) come:

$$z_k^{(l)} = \sum_{i=1}^{n_{l-1}} W_{ki}^{(l)} a_i^{(l-1)} + b_k^{(l)},$$

dove $a_i^{(l-1)}$ rappresenta l'uscita del neurone i-esimo dello layer precedente, mentre $z_k^{(l)}$ è l'ingresso netto al neurone corrente.

In forma vettoriale, l'espressione precedente diventa:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}.$$

Per introdurre la non linearità — fondamentale per permettere alla rete di apprendere relazioni complesse — si applica la funzione di attivazione:

$$\mathbf{a}^{(l)} = f^{(l)}\left(\mathbf{z}^{(l)}\right),$$

dove $\mathbf{a}^{(l)}$ è il vettore delle uscite dello layer l.

Senza la funzione di attivazione non lineare, la rete si ridurrebbe a una composizione di trasformazioni lineari, quindi a una singola trasformazione affine, incapace di modellare comportamenti complessi.

L'output di un layer diventa l'input del successivo, propagando l'informazione attraverso la rete fino all'output finale. Per l'input layer si pone:

$$\mathbf{a}^{(0)} = \mathbf{x},$$

e, in conclusione, l'output complessivo della rete è:

$$\mathbf{y} = \mathbf{a}^{(L)}.$$

L'output finale della rete viene confrontato con il valore atteso (target) tramite una funzione obiettivo (loss function), che quantifica l'errore della predizione. La scelta della funzione dipende dal compito: ad esempio, l'errore quadratico medio è usato in regressione, mentre l'entropia incrociata è tipicamente usata nei problemi di classificazione.

Per ridurre tale errore si utilizza l'algoritmo di retropropagazione (backpropagation), che si articola in tre fasi principali:

1. **Forward pass**: calcolo delle uscite della rete dato l'input;
2. **Backward pass**: calcolo del gradiente della funzione obiettivo rispetto ai pesi mediante la regola della catena;
3. **Aggiornamento**: modifica dei pesi (tipicamente tramite discesa del gradiente) per ridurre progressivamente l'errore.

L'addestramento consiste nella ripetizione iterativa di questo ciclo fino a quando la rete non riesce ad apprendere una rappresentazione adeguata dei dati e a generalizzare su nuovi esempi.

Un aspetto cruciale del forward pass è rappresentato dalla funzione di attivazione $f^{(l)}(\cdot)$, la quale introduce la non linearità nel modello. Nel paragrafo seguente analizzeremmo in dettaglio le principali funzioni di attivazione e il loro ruolo nel comportamento della rete neurale.

4.7 Funzioni di attivazione

Le funzioni di attivazione rappresentano uno degli elementi più importanti del Multi Layer Perceptron. Esse introducono la non linearità nel modello, consentendo alla rete neurale di approssimare relazioni complesse tra input e output. Senza di esse, anche una rete con molti layer sarebbe equivalente a un'unica trasformazione lineare.

Consideriamo, ad esempio, una rete neurale costituita da due layer, entrambi con funzioni di attivazione lineari:

1. il primo layer calcola
$$z_1 = \omega_1 x + b_1;$$
2. il secondo layer calcola
$$z_2 = \omega_2 z_1 + b_2.$$

Sostituendo la prima equazione nella seconda si ottiene:

$$z_2 = \omega_2(\omega_1 x + b_1) + b_2 = (\omega_1\omega_2)x + (\omega_2 b_1 + b_2) = w'x + b'.$$

La rete a due layer si riduce quindi a una singola trasformazione lineare. In generale, qualunque rete composta da layer lineari rimane equivalente a una sola trasformazione affine: l'aggiunta di layer non aumenta la complessità del modello.

Le funzioni di attivazione non lineari superano questa limitazione, permettendo alla rete di costruire rappresentazioni gerarchiche e di apprendere relazioni non lineari nei dati.

Un altro aspetto importante delle funzioni di attivazione è che ci consentono di mappare un flusso di input di distribuzione sconosciuta e di ridimensionarlo a uno noto (ad esempio, la funzione sigmoide mappa qualsiasi input a un valore compreso tra 0 e 1). Ciò rende la rete capace di modellare fenomeni complessi e di risolvere problemi che altrimenti sarebbero irraggiungibili, quali:

1. Apprendere modelli complessi: i dati del mondo reale raramente seguono modelli lineari semplici. Le funzioni di attivazione non lineari consentono alle reti di approssimare qualsiasi funzione complessa, rendendole approssimatori di funzioni universali.
2. Creare gerarchie di caratteristiche: nelle reti profonde, ogni livello può apprendere caratteristiche sempre più sofisticate perché le funzioni non lineari consentono la combinazione di caratteristiche più semplici in modi non lineari.
3. Catturare relazioni non lineari: che si tratti di riconoscere volti nelle immagini o di prevedere i prezzi delle azioni, la maggior parte dei problemi del mondo reale comporta relazioni non lineari che richiedono funzioni non lineari per essere modellate in modo efficace.

4.8 Differenziabilità e problemi di gradiente

Affinché la backpropagation sia applicabile, la funzione di attivazione deve essere differenziabile quasi ovunque. Tuttavia, la derivabilità non è solo un requisito tecnico: la forma della derivata influisce direttamente sulla stabilità del gradiente e sulla velocità di convergenza. Funzioni troppo piatte (es. sigmoide) portano al problema di vanishing gradient, mentre funzioni non limitate (es. ReLU) possono causare l'opposto problema di exploding gradient.

4.8.1 Funzione a soglia (o di Heaviside)

Storicamente la prima funzione di attivazione era la funzione a soglia:

$$H(z) = \begin{cases} 1 & \text{se } z \geq 0, \\ 0 & \text{se } z < 0. \end{cases}$$

Oggi però è poco comune perché non è adatta all'ottimizzazione tramite gradienti. Infatti, la funzione di Heaviside $H(z)$ ha una discontinuità a salto nell'origine, e dunque non può essere derivabile in senso classico. Tuttavia, è noto che in senso distribuzionale ammette una derivata debole, che è la famosa distribuzione δ di Dirac. Tali argomenti esulano però dallo scopo di questa tesi.

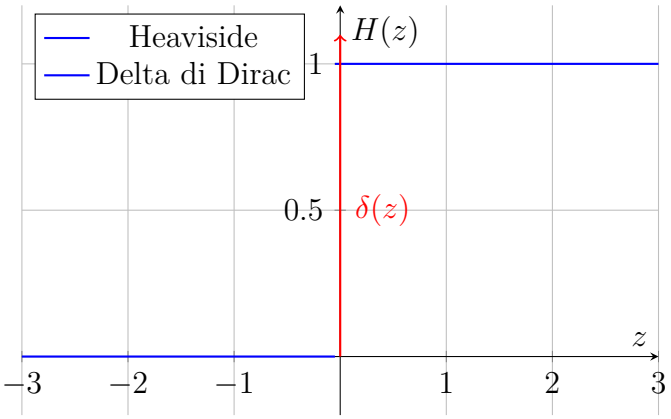


Figura 4.3: Funzione a soglia e derivata (impulso di Dirac)

Affinché una rete neurale possa essere addestrata tramite backpropagation, tutti i suoi componenti devono essere differenziabili, inclusa la funzione di attivazione. Durante la fase di retropropagazione, infatti, l'algoritmo calcola il gradiente della funzione di perdita rispetto ai pesi utilizzando la regola della catena. Per questo motivo, la scelta di funzioni di attivazione con derivate ben comportate è cruciale per garantire la stabilità dell'apprendimento.

Tra le funzioni di attivazione più comuni troviamo quelle che illustriamo nelle prossime sottosezioni.

4.8.2 Sigmoide

La funzione sigmoide è una funzione di attivazione non lineare molto diffusa che mappa i dati di input nell'intervallo di output $(0, 1)$. A differenza della funzione gradino, la funzione sigmoide non produce solo 0 o 1, ma numeri compresi in quell'intervallo (esclusi 0 e 1). La funzione è

$$\sigma(z) = \frac{1}{1 + e^{-z}},$$

e la sua derivata è

$$\sigma'(z) = \frac{e^{-z}}{(1 - e^{-z})^2} = f(z) \cdot (1 - f(z))$$

4.8.3 Tangente iperbolica

La tangente iperbolica, simile alla sigmoide, mappa i dati di input su una curva a forma di S, ma in questo caso l'intervallo di output è $(-1, 1)$ ed è centrato in 0. La formula della tangente

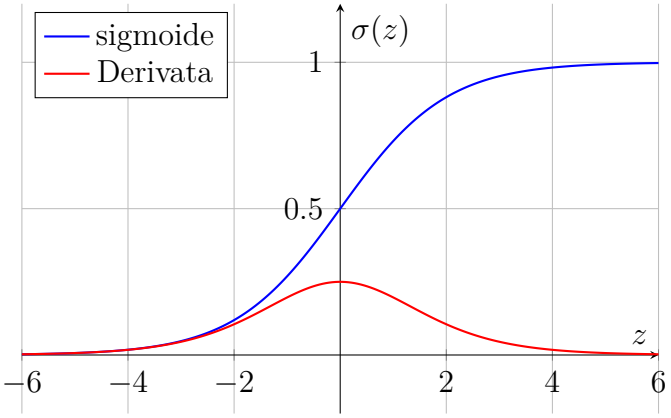


Figura 4.4: Funzione sigmoide e derivata

iperbolica è:

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{\sinh(z)}{\cosh(z)},$$

e ha derivata

$$f'(z) = 1 - f(z)^2 = 1 - \tanh^2(z) = \frac{1}{\cosh^2(z)}.$$

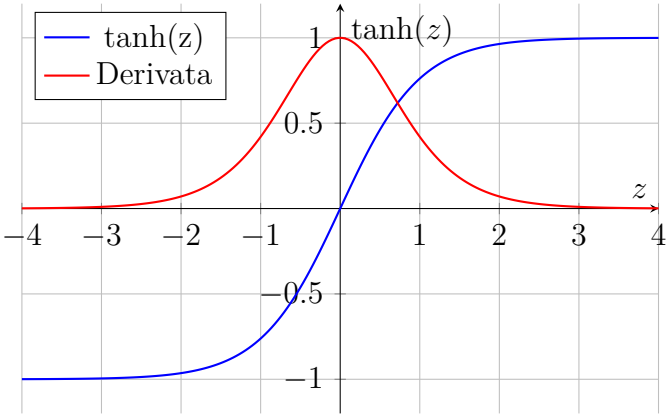


Figura 4.5: Tangente iperbolica e derivata

4.8.4 Tangente inversa

La funzione tangente inversa $\tan^{-1}$ (altrimenti detta arcotangente) è un'altra funzione di attivazione non lineare. Simile alla funzione sigmoide e alla funzione tanh, ha una forma a "S" e anche le derivate hanno forme simili. La funzione tangente inversa produce valori compresi nell'intervallo $(-\pi/2, \pi/2)$. La sua formula è

$$f(z) = \tan^{-1}(z),$$

e la sua derivata è

$$f'(z) = \frac{1}{1 + z^2}.$$

Un grande vantaggio dell'arcotangente come funzione di attivazione è che la sua derivata non contiene termini esponenziali. Ciò rende il calcolo più semplice e veloce rispetto ad altre funzioni, come la sigmoide o la tangente iperbolica.

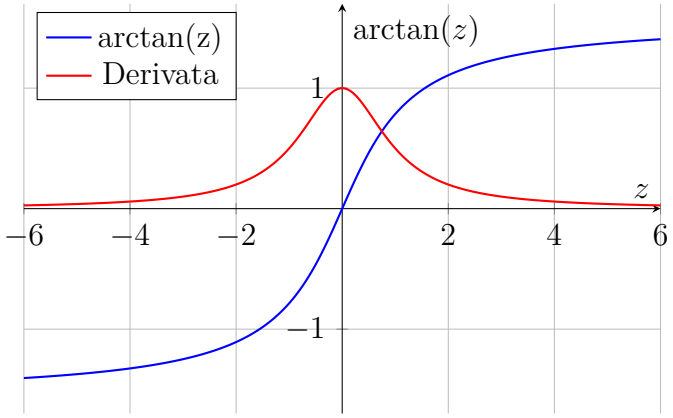


Figura 4.6: Tangente inversa e derivata

4.8.5 Vanishing ed Exploding Gradient

Definizione 4.8.1 (Rete neurale profonda)**.** Una rete neurale profonda (o deep neural network) è una funzione

$$F : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$$

ottenuta come composizione di L trasformazioni parametriche, ciascuna associata a un layer della rete. Più precisamente, per ogni $l = 1, \dots, L$ definiamo

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \qquad \mathbf{a}^{(l)} = f^{(l)}(\mathbf{z}^{(l)}),$$

dove:

- $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ è la matrice dei pesi del layer l;
- $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$ è il vettore dei bias;
- $f^{(l)} : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$ è una funzione di attivazione applicata elemento per elemento;
- $\mathbf{a}^{(0)} = \mathbf{x} \in \mathbb{R}^{n_0}$ è il vettore di input.

La rete neurale profonda è quindi la mappa

$$F(\mathbf{x}) = \mathbf{a}^{(L)} = f^{(L)}\Big(\mathbf{W}^{(L)} f^{(L-1)}\Big(\cdots f^{(1)}\Big(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\Big)\cdots\Big) + \mathbf{b}^{(L)}\Big).$$

La rete si dice profonda quando il numero dei layer nascosti, cioè $L-1$, è maggiore o uguale a 2. Il vettore di tutti i parametri della rete è

$$\theta = \Big\{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \dots, \mathbf{W}^{(L)}, \mathbf{b}^{(L)}\Big\}.$$

Nell'addestramento delle reti neurali profonde, ci sono due problematiche comuni note come vanishing gradient ed exploding gradient. Entrambe derivano dal comportamento delle derivate delle funzioni di attivazione e dalla propagazione del gradiente all'indietro tramite la regola della catena.

Durante la backpropagation, il gradiente dell'errore rispetto ai pesi di un layer dipende dalla derivata della funzione di attivazione e dal gradiente proveniente dallo layer successivo. Formalmente, se C è la funzione di perdita e $a^{(l)}$ l'uscita dello layer l, allora:

$$\frac{\partial C}{\partial a^{(l)}} = \frac{\partial C}{\partial a^{(L)}} \prod_{k=l}^{L-1} \frac{\partial a^{(k+1)}}{\partial a^{(k)}}.$$

Il gradiente che raggiunge gli layer iniziali è quindi il risultato di un prodotto di molte derivate. Se queste derivate hanno valori tipicamente minori di 1, il prodotto tenderà a zero (vanishing); se invece sono maggiori di 1, tenderà a crescere senza controllo (exploding).

Vanishing gradient Il vanishing gradient si verifica quando le derivate delle funzioni di attivazione hanno valori strettamente minori di 1 (ad esempio sigmoide o tanh nelle loro zone di saturazione). In reti profonde, il prodotto di molti termini < 1 tende esponenzialmente a zero:

$$\prod_{k=1}^{n} \sigma'(z^{(k)}) \ \to \ 0 \quad \text{per} \quad n \to +\infty.$$

Questo riduce drasticamente l'aggiornamento dei pesi nei layer iniziali, rallentando o bloccando l'apprendimento.

Exploding gradient L'exploding gradient, al contrario, si manifesta quando le derivate (o i pesi stessi) assumono valori > 1. In questo caso il prodotto tende a crescere molto rapidamente:

$$\prod_{k=1}^{n} \sigma'(z^{(k)}) \ \to \ +\infty \quad \text{per} \quad n \to +\infty.$$

Ciò causa gradienti enormi che portano a instabilità numerica, oscillazioni nei valori dei pesi e potenziale divergenza dell'ottimizzazione. Questo fenomeno si verifica più facilmente nelle reti molto profonde o nelle reti ricorrenti (RNN), dove i gradienti vengono propagati attraverso molti layer o passi temporali, amplificandosi progressivamente. Entrambi i fenomeni ostacolano l'addestramento di reti profonde. Per questo spesso si usano funzioni di attivazione non saturanti come la ReLU, che mantengono derivate costanti o circa uguali a 1.

4.8.6 ReLU (Rectified Linear Unit)

La funzione ReLU è definita come

$$f(z) = \max(0, z),$$

cioè

$$f(z) = \begin{cases} 0 & \text{se } z \leq 0, \\ z & \text{se } z > 0. \end{cases}$$

In altre parole, se l'input z è negativo, la ReLU azzera e il neurone non si attiva. Altrimenti, se l'input è positivo, la ReLU lo lascia invariato e il neurone si attiva in proporzione alla forza del segnale. In sostanza, la ReLU taglia via la parte negativa e lascia passare solo i valori positivi.

La derivata della ReLU è molto semplice

$$f'(z) = H(z) = \begin{cases} 0 & \text{se } z \leq 0, \\ 1 & \text{se } z > 0, \end{cases}$$

dove $H(z)$ è la funzione di Heaviside, introdotta in precedenza. Il valore per $z = 0$ di solito si definisce arbitrariamente, perché è un punto di non derivabilità, ma in pratica non influisce molto.

La Rectified Linear Unit (ReLU) è la funzione di attivazione più utilizzata nelle reti neurali profonde per la sua semplicità e l'efficienza computazionale. Rispetto alla sigmoide e alla tangente iperbolica, favorisce un addestramento più rapido poiché non satura per valori positivi (cioè danno lo stesso valore in uscita per valori grandi) e la derivata rimane costante ($f'(z) = 1$ per $z > 0$), riducendo il problema del gradiente vanishing. Il principale limite è che per $z < 0$ l'uscita e la derivata si annullano, causando talvolta i cosiddetti neuroni morti. Per attenuare questo effetto sono state introdotte varianti come la Leaky ReLU, data da

$$f(z) = \begin{cases} 0.01z & \text{se } z \leq 0, \\ z & \text{se } z > 0, \end{cases}$$

che lascia quindi passare un piccolo flusso anche per $z < 0$, e la Parametric ReLU, data da

$$f(z) = \begin{cases} \eta z & \text{se } z \leq 0, \\ z & \text{se } z > 0, \end{cases}$$

dove η è un parametro che può essere appreso durante il training. Tipicamente viene inizializzato con un piccolo valore positivo, come per esempio $\eta = 0.01$.

Ogni funzione di attivazione ha vantaggi e limiti: la scelta dipende dal problema da affrontare e dall'architettura della rete. La loro introduzione ha rappresentato un punto di svolta nello sviluppo delle reti neurali, consentendo di superare i limiti dei modelli puramente lineari.

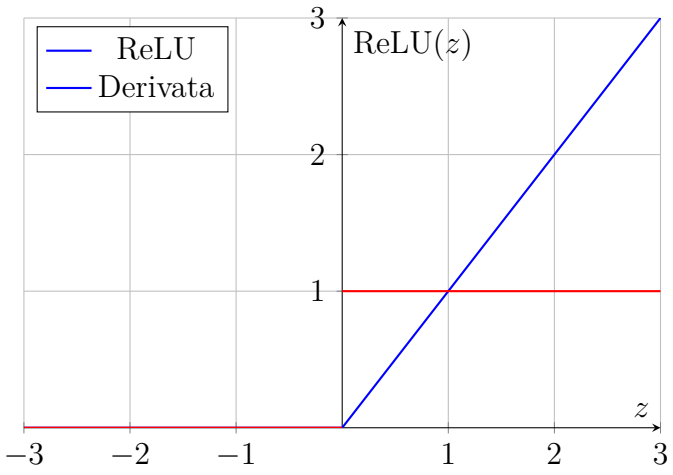


Figura 4.7: ReLU e derivata ($H(z)$)

In sintesi, le funzioni di attivazione determinano il comportamento non lineare della rete e influenzano in modo significativo la velocità e la stabilità dell'apprendimento. La scelta della funzione più adatta dipende dal tipo di problema, dalla profondità della rete e dalla natura dei dati in ingresso. Nelle sezioni successive vedremo come, durante l'addestramento, la retropropagazione sfrutta le derivate di queste funzioni per aggiornare i pesi della rete.

Vediamo un esempio per chiarire i concetti precedenti.

Esempio 4.8.2. Supponiamo di avere una rete neurale con 2 neuroni nell'input layer, 2 nell'unico hidden layer e un neurone nell'output layer. Ci vengono inoltre forniti i seguenti dati:

- il vettore in ingresso
$$\mathbf{x} = \begin{pmatrix} 1 \\ 2 \end{pmatrix},$$
- la matrice pesi del hidden layer
$$\mathbf{W}^{(1)} = \begin{bmatrix} 0.1 & 0.3 \\ 0.4 & 0.6 \end{bmatrix},$$
- il vettore bias del hidden layer
$$\mathbf{b}^{(1)} = \begin{pmatrix} 0.5 \\ 0.2 \end{pmatrix},$$

- la matrice pesi del output layer

$$\mathbf{W}^{(2)} = \begin{bmatrix} 0.7 & 0.9 \end{bmatrix},$$

- il vettore bias del output layer

$$\mathbf{b}^{(2)} = \begin{pmatrix} 0.1 \end{pmatrix},$$

- nel primo hidden layer la funzione di attivazione è una ReLU, e nel secondo una sigmoide.

Vogliamo calcolare il valore in uscita.

Cominciamo con il primo hidden layer

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)} = \begin{bmatrix} 0.1 & 0.3 \\ 0.4 & 0.6 \end{bmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} + \begin{pmatrix} 0.5 \\ 0.2 \end{pmatrix} = \begin{pmatrix} 1.2 \\ 1.8 \end{pmatrix}.$$

Per ottenere i valori in uscita dei neuroni applichiamo la ReLU:

$$f(\mathbf{z}^{(1)}) = \max(0, \mathbf{a}^{(1)}) = \begin{pmatrix} 1.2 \\ 1.8 \end{pmatrix} = \mathbf{a}^{(1)}.$$

Nello stesso modo otteniamo per l'output layer:

$$\mathbf{z}^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)} = \begin{bmatrix} 0.7 & 0.9 \end{bmatrix} \begin{pmatrix} 1.2 \\ 1.8 \end{pmatrix} + 0.1 = 2.68.$$

Per ottenere il valore $\mathbf{a}^{(L)}$ diamo questo valore in ingresso alla sigmoide:

$$\sigma(\mathbf{z}^{(1)}) = \frac{1}{1 + e^{-\mathbf{z}^{(1)}}} = \frac{1}{1 + e^{-2.68}} = 0.9358.$$

4.9 Funzione obiettivo

La funzione obiettivo (nota anche come funzione di perdita o loss function) è una funzione scalare che misura quanto l'uscita della rete si discosti dal valore atteso (target).

Formalmente, se la rete calcola una predizione $\mathbf{y}$ per un certo input $\mathbf{x}$, e il valore reale (etichetta) è $\hat{\mathbf{y}}$, allora la funzione obiettivo è

$$C(\mathbf{y}, \hat{\mathbf{y}}) = \text{errore tra il valore predetto è quello reale.}$$

L'addestramento di una rete neurale consiste nel trovare i parametri che minimizzano questa funzione rispetto ai propri parametri.

Anche se la funzione obiettivo $C(\mathbf{y}, \hat{\mathbf{y}})$ sembra dipendere solo da $\mathbf{y}$, in realtà la dipendenza da tutti i pesi e bias è implicata, perché

$$\mathbf{y} = f(\mathbf{z}^{(L)}) = \mathbf{a}^{(L)},$$

dove ogni output $a_j^{(L)}$ dipende dalle uscite degli layer precedenti, che a loro volta dipendono dai pesi e bias tramite il forward pass. Ricordiamo che:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)},$$
$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)}).$$

Quindi, anche se scriviamo $C(\mathbf{y}, \hat{\mathbf{y}})$, in realtà abbiamo

$$C = C(\mathbf{y}, \mathbf{a}^{(L)}(\mathbf{z}^{(L)}(\dots \mathbf{W}^{(l)}, \mathbf{b}^{(l)}, \dots).$$

Riassumendo, possiamo quindi dire che ogni peso e ogni bias influenzano l'output finale, e quindi anche la funzione obiettivo. Nell'ultimo capitolo vedremmo la backpropagation che serve proprio a calcolare quanto ogni peso contribuisce all'errore totale, cioè le derivate

$$\frac{\partial C}{\partial \omega_{ij}^{(l)}}, \frac{\partial C}{\partial b_i^{(l)}}.$$

Esistono diverse funzioni obiettivo, ciascuna adatta a differenti compiti di apprendimento. In generale, tutte operano quantificando la differenza tra le uscite previste dal modello e i valori target. Tra le più utilizzate vi sono:

- l'**errore quadratico medio** (Mean Squared Error, MSE), tipico dei problemi di regressione;
- l'**entropia incrociata** (Cross-Entropy Loss), comune nei problemi di classificazione.

4.9.1 Errore quadratico medio (MSE)

Il MSE viene utilizzato nei problemi di regressione, in cui il modello deve stimare un valore numerico continuo. Dal punto di vista matematico, un modello di regressione apprende una funzione del tipo:

$$f : \mathbb{R}^n \to \mathbb{R}.$$

La funzione MSE quantifica l'errore come media del quadrato delle differenze tra i valori previsti e i target effettivi:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

dove y_i indica il valore reale e $\hat{y}_i$ la previsione del modello per l'osservazione i-esima.

Minimizzare l'MSE equivale quindi a ridurre la distanza media (al quadrato) tra le previsioni e i valori attesi.

4.9.2 Entropia incrociata

Quando il problema di apprendimento riguarda la classificazione, ossia l'assegnazione di un campione ad una delle $K \in \mathbb{N}$ classi possibili, si utilizza tipicamente come funzione obiettivo l'entropia incrociata. Dal punto di vista matematico, il modello approssima una funzione

$$f : \mathbb{R}^n \to \{1, 2, \dots, K\},$$

che associa ad un vettore di ingresso una delle K etichette.

L'entropia incrociata misura la distanza tra la distribuzione vera (rappresentata dall'etichetta) e quella predetta dal modello. Nel caso di K classi, essa è definita come

$$L = -\sum_{k=1}^{K} y_k \log(p_k),$$

dove:

- y_k è l'etichetta vera espressa in one-hot encoding, quindi $y_k = 1$ solo per la classe corretta e $y_k = 0$ per tutte le altre;

- p_k è la probabilità predetta dal modello per la classe k.

Poiché $y_k = 0$ per tutte le classi tranne quella giusta, la formula si riduce a

$$L = -\log(p_{\text{classe corretta}}).$$

In questo modo, la funzione di costo penalizza fortemente i casi in cui il modello assegna una bassa probabilità alla classe corretta, mentre restituisce valori piccoli (quindi migliori) quando la probabilità predetta per la classe vera è alta.

Esempio 4.9.1. Consideriamo $K = 3$ classi e supponiamo che la classe corretta sia la 2, quindi $\mathbf{y} = (0, 1, 0)$. Se il modello produce

$$p = \begin{pmatrix} 0.2 & 0.7 & 0.1 \end{pmatrix},$$

allora

$$L = -\log(0.7) \approx 0.357,$$

cioè un errore relativamente basso, perché la probabilità assegnata alla classe corretta è alta (70%).

Se invece il modello predice

$$p = \begin{pmatrix} 0.8 & 0.1 & 0.1 \end{pmatrix},$$

otteniamo

$$L = -\log(0.1) \approx 2.302,$$

un errore molto più grande, poiché la classe corretta ha ricevuto solo il 10% di probabilità.

Nei capitoli successivi approfondiremo come, tramite la backpropagation e la discesa del gradiente, la rete possa apprendere autonomamente i propri parametri per minimizzare la funzione obiettivo.

Capitolo 5

Discesa del Gradiente

5.1 Introduzione

Le reti neurali sono sistemi complessi il cui comportamento è determinato da numerosi parametri: pesi (weights) e bias. All'inizio del processo di addestramento, questi parametri vengono tipicamente inizializzati con valori casuali, risultando in prestazioni scadenti della rete.

La sfida fondamentale dell'apprendimento automatico consiste nel trovare un metodo matematico per:

- **Valutare** le prestazioni della rete;
- **Comunicare** alla rete come migliorare le sue prestazioni;
- **Aggiornare** i parametri in modo sistematico.

Per affrontare questa sfida, consideriamo come funzione obiettivo (o loss function) l'errore quadratico medio (MSE). Come abbiamo visto nel capitolo precedente, questa funzione prende in input i valori dell'output layer e restituisce un valore scalare che quantifica quanto l'output della rete si discosta dall'output desiderato. Assume quindi valori piccoli quando l'output è buono e valori grandi quando l'output è scorretto.

Il nostro obiettivo diventa quindi trovare i parametri della rete che minimizzano questa funzione obiettivo. Mentre in teoria potremmo risolvere il problema algebricamente ponendo il gradiente uguale a zero, la realtà risulta più complicata, per vari motivi:

- le funzioni obiettivo delle reti neurali sono altamente non lineari e non convesse,
- il numero di parametri può essere nell'ordine dei milioni,
- una soluzione analitica è computazionalmente intrattabile.

È in questo contesto che la discesa del gradiente emerge come strumento fondamentale per l'ottimizzazione numerica.

5.2 Fondamenti Matematici della Discesa del Gradiente

5.2.1 La direzione di massima discesa

Teorema 5.2.1 (**Direzione di Massima Salita**). *Sia $\Omega \subseteq \mathbb{R}^n$ aperto e sia $f : \Omega \to \mathbb{R}$ differenziabile in $\mathbf{x}_0 \in \Omega$ con $\nabla f(\mathbf{x}_0) \neq 0$. Allora, tra tutti i vettori unitari $\mathbf{d} \in \mathbb{R}^n$ con $\|\mathbf{d}\| = 1$, la direzione in cui la derivata direzionale di f in $\mathbf{x}_0$, $\partial_d f(x_0)$, è massima è data da:*

$$\mathbf{d}^* = \frac{\nabla f(\mathbf{x}_0)}{\|\nabla f(\mathbf{x}_0)\|}.$$

Dimostrazione. Per il Teorema 2.3.11, la derivata direzionale di f nella direzione $\mathbf{d}$ è:

$$\partial_d f(\mathbf{x}_0) = \nabla f(\mathbf{x}_0) \cdot \mathbf{d}.$$

Per la disuguaglianza di Cauchy-Schwarz:

$$\nabla f(\mathbf{x}_0) \cdot \mathbf{d} \leq \|\nabla f(\mathbf{x}_0)\|\|\mathbf{d}\| = \|\nabla f(\mathbf{x}_0)\|.$$

L'uguaglianza si verifica se e solo se $\mathbf{d}$ è parallelo a $\nabla f(\mathbf{x}_0)$, cioè:

$$\mathbf{d} = \frac{\nabla f(\mathbf{x}_0)}{\|\nabla f(\mathbf{x}_0)\|}.$$

□

Osservazione 5.2.2 (**Direzione di Massima Discesa**)**.** Poiché il gradiente $\nabla f(\mathbf{x}_0)$ indica la direzione di massima crescita, la direzione opposta,

$$-\mathbf{d}^* = -\frac{\nabla f(\mathbf{x}_0)}{\|\nabla f(\mathbf{x}_0)\|},$$

è la direzione di discesa più rapida.

Dopo aver visto questa proprietà del gradiente, possiamo ora formalizzare l'algoritmo che sfrutta tali proprietà per l'ottimizzazione.

5.2.2 Derivazione dell'Algoritmo di Discesa del Gradiente

Partendo da un punto iniziale $\mathbf{x}_0$, consideriamo lo sviluppo di Taylor al primo ordine di f attorno a $\mathbf{x}_0$:

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) + o(\|\mathbf{x} - \mathbf{x}_0\|).$$

Consideriamo ora il punto:

$$\mathbf{x} = \mathbf{x}_0 - \alpha \nabla f(\mathbf{x}_0),$$

dove $\alpha > 0$ è un parametro piccolo chiamato "learning rate" o dimensione del passo.
Assumiamo inoltre che:

1. α è sufficientemente piccolo per poter trascurare i termini dello sviluppo di Taylor ordine quadratico e maggiore;
2. $\nabla f(\mathbf{x}_0) \neq 0$.

Sostituendo il punto nell'espansione di Taylor otteniamo

$$\begin{aligned} f(\mathbf{x}_0 - \alpha \nabla f(\mathbf{x}_0)) &= f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0) \cdot (-\alpha \nabla f(\mathbf{x}_0)) + o(\alpha) \\ &= f(\mathbf{x}_0) - \alpha \|\nabla f(\mathbf{x}_0)\|^2 + o(\alpha) \end{aligned}$$

Dato che α è positivo e il termine $o(\alpha)$ è trascurabile, sappiamo con certezza che, se da $\mathbf{x}_0$ ci muoviamo di α nella direzione $-\nabla f(\mathbf{x}_0)$, otteniamo un valore più basso, cioè

$$f(\mathbf{x}_0 - \alpha \nabla f(\mathbf{x}_0)) < f(\mathbf{x}_0).$$

Questo dimostra che, muovendoci nella direzione opposta al gradiente, otteniamo un miglioramento della funzione obiettivo.

5.3 Algoritmo di Discesa del Gradiente

Definizione 5.3.1 (**Algoritmo di Discesa del Gradiente**)**.** Dato un punto iniziale $\mathbf{x}_0 \in \Omega$ e una funzione obiettivo $f \in C^1(\Omega)$, l'algoritmo di discesa del gradiente procede iterativamente come segue:

1. **Inizializzazione**: Si scelgono un punto $\mathbf{x}_0$ e una learning rate $\alpha > 0$;
2. **Iterazione** per $k = 0, 1, 2, \ldots$:
   (a) si calcola il gradiente: $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$;
   (b) si aggiorna la posizione: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \mathbf{g}_k$;
   (c) si verifica il criterio di arresto (che verrà trattato in più dettaglio nella Sezione 5.3.2).
3. **Fino a che** il criterio di arresto è soddisfatto.

Osservazione 5.3.2. L'algoritmo della discesa del gradiente può essere espresso mediante la seguente formula iterativa:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k), \tag{5.3.1}$$

dove $\alpha_k > 0$ rappresenta la learning rate all'k-esima iterazione. Tale parametro può essere scelto come costante, cioè $\alpha_k = \alpha$ per ogni k, oppure come variabile, determinato a ogni iterazione in base a specifiche strategie di ottimizzazione, come la line search. L'equazione (5.3.1) descrive quindi l'aggiornamento successivo del punto $\mathbf{x}_k$ nella direzione opposta al gradiente di f in $\mathbf{x}_k$, ossia nella direzione di massima discesa.

5.3.1 Scelta del Learning Rate

La scelta del learning rate α è cruciale per le prestazioni dell'algoritmo. Infatti, la casistica si può riassumere nel seguente modo:

- α **troppo piccolo**:
  - Convergenza lenta
  - Maggior numero di iterazioni necessarie
  - Rischio di rimanere intrappolati in minimi locali
- α **troppo grande**:
  - Possibili oscillazioni attorno al minimo
  - Rischio di divergenza
  - Nessuna garanzia di convergenza
- α **ottimale**:
  - Bilancio tra velocità e stabilità
  - Dipendenza dalla funzione specifica
  - Spesso determinato empiricamente

5.3.2 Criterio di arresto

Nella Definizione 5.3.1 abbiamo detto che aggiorniamo la posizione affinché il criterio di arresto non è soddisfatto. Non abbiamo ancora stabilito quale sia tale criterio. Una prima idea potrebbe essere di usare la condizione che ci indica di aver raggiunto esattamente il minimo, cioè $\nabla f(\mathbf{x}_k) = 0$ come criterio di arresto. Tuttavia, questa condizione non è direttamente applicabile, poiché il calcolo numerico del gradiente raramente sarà identico a zero.

In pratica usiamo come criterio di arresto che consiste nel verificare se la norma del gradiente $\|\nabla f(\mathbf{x}_k)\|$ è inferiore a una soglia prestabilita, nel qual caso si interrompe il processo.

In alternativa, è possibile calcolare la differenza assoluta tra i valori della funzione obiettivo per ogni due iterazioni successive

$$|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| < \varepsilon,$$

e, se la differenza è inferiore a una soglia prestabilita $\varepsilon > 0$, interrompere il processo.

Un altro modo ancora sarebbe calcolare la norma $\|\mathbf{x}_{k+1} - \mathbf{x_k}\|$ della differenza tra due iterazioni successive, e fermarci se la norma è minore di una soglia prestabilita ε:

$$\|\mathbf{x}_{k+1} - \mathbf{x_k}\| < \varepsilon.$$

Spesso conviene calcolare i valori relativi delle quantità precedenti per renderli indipendenti dalla scala. Cioè:

$$\frac{|f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)|}{|f(\mathbf{x}_k)|} < \varepsilon,$$

e

$$\frac{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|}{\|\mathbf{x}_k\|} < \varepsilon.$$

Tutti questi criteri di arresto riflettono, in modi diversi, la stessa idea di fondo: che la successione $\{\mathbf{x}_k\}$ stia convergendo verso un punto stazionario. In altre parole, man mano che il numero di iterazioni cresce, le variazioni della posizione $\mathbf{x}_k$ o dei valori della funzione obiettivo $f(\mathbf{x}_k)$ diventano sempre più piccole, indicando che il processo di ottimizzazione si sta stabilizzando e che ulteriori aggiornamenti non porterebbero cambiamenti significativi.

5.4 Metodo della Discesa più Rapida

In questa tipologia della discesa del gradiente scegliamo la dimensione del passo α non costante, ma in modo tale per ottenere la massima riduzione della funzione obiettivo ad ogni singolo passo.

Definizione 5.4.1 (**Algoritmo della discesa più rapida**)**.** Nell'algoritmo della discesa più rapida (o steepest gradient descent) il punto successivo $\mathbf{x}_{k+1}$ si ottiene a partire dal punto corrente $\mathbf{x}_k$ secondo la formula iterativa

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k), \tag{5.4.1}$$

dove $\alpha_k \geq 0$ rappresenta il tasso di apprendimento (o learning rate) della k-esima iterazione. Il valore di α_k viene determinato come il minimo della funzione monodimensionale

$$\phi_k(\alpha) = f\Big(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)\Big), \tag{5.4.2}$$

ossia

$$\alpha_k = \arg\min_{\alpha \geq 0} f\Big(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)\Big). \tag{5.4.3}$$

In tal modo, ad ogni iterazione, si sceglie il passo ottimale che garantisce la massima riduzione della funzione obiettivo lungo la direzione di massima discesa individuata dal gradiente.

Osservazione 5.4.2 (**Punti stazionari**). Se per un certo indice k si ha $\nabla f(\mathbf{x}_k) = 0$, allora, per il criterio di Fermat, $\mathbf{x}_k$ è un punto stazionario della funzione f. In tal caso, la formula iterativa dell'algoritmo (5.4.1) implica che

$$\mathbf{x}_{k+1} = \mathbf{x}_k,$$

e quindi il processo si arresta, non essendoci più una direzione di discesa lungo cui muoversi.

Osserviamo che questo metodo fa passi ortogonali, fintantoché non si arresta in un punto stazionario.

Proposizione 5.4.3. *Sia $\{\mathbf{x}_k\}_{k=0}^{\infty}$ una sequenza generata dall'algoritmo della discesa più ripida applicato a una funzione $f \in C^1(\Omega)$. Se per ogni $k \in \mathbb{N}$ risulta $\nabla f(\mathbf{x}_k) \neq \mathbf{0}$, allora i vettori $\mathbf{x}_{k+1} - \mathbf{x}_k$ e $\mathbf{x}_{k+2} - \mathbf{x}_{k+1}$ sono non nulli e ortogonali per ogni $k \in \mathbb{N}$.*

Dimostrazione. Sfruttando la formula iterativa della discesa più rapida (5.4.1), calcoliamo quindi il prodotto scalare dei due vettori $\mathbf{x}_{k+1} - \mathbf{x}_k$ e $\mathbf{x}_{k+2} - \mathbf{x}_{k+1}$:

$$(\mathbf{x}_{k+1} - \mathbf{x}_k) \cdot (\mathbf{x}_{k+2} - \mathbf{x}_{k+1}) = \alpha_k \alpha_{k+1} [\nabla f(\mathbf{x}_k) \cdot \nabla f(\mathbf{x}_{k+1})].$$

Il prodotto scalare è zero se e solo se uno dei due vettori è zero, oppure se i due vettori sono ortogonali. Per ipotesi $\nabla f(\mathbf{x}_k)$ e $\nabla f(\mathbf{x}_{k+1})$ sono non nulli, e dunque, per l'Osservazione 5.4.2, notiamo che $\mathbf{x}_{k+1} - \mathbf{x}_k \neq 0$ e $\mathbf{x}_{k+2} - \mathbf{x}_{k+1} \neq 0$. Quindi basta dimostrare che

$$\nabla f(\mathbf{x}_k) \cdot \nabla f(\mathbf{x}_{k+1}) = 0.$$

Per fare ciò, osserviamo che α_k è un scalare non negativo che abbiamo ottenuto calcolando il minimo della funzione $\phi_k(\alpha) = f\Big(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)\Big)$. Quindi, per il teorema di Fermat, dobbiamo avere

$$\phi_k'(\alpha) = 0,$$

e, applicando la regola della catena, vediamo che

$$0 = \phi_k'(\alpha_k) = \nabla f\Big(\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)\Big) \cdot \Big(-\nabla f(\mathbf{x}_k)\Big) = -\nabla f(\mathbf{x}_{k+1}) \cdot \nabla f(\mathbf{x}_k).$$

□

La seguente proposizione garantisce che per ogni nuovo punto non stazionario generato dall'algoritmo il rispettivo valore della funzione f diminuisce in valore.

Proposizione 5.4.4 (**Proprietà di Decrescita**). *Se $\{\mathbf{x}_k\}_{k=0}^{\infty}$ è la sequenza dell'algoritmo della discesa più rapida per la funzione $f \in C^1(\Omega)$, e se $\nabla f(\mathbf{x}_k) \neq 0$ per qualche $k \in \mathbb{N}$, allora $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$.*

Dimostrazione. Nell'algoritmo della discesa più rapida otteniamo il punto successivo a $\mathbf{x}_k$ applicando la formula (5.4.1), cioè

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k),$$

dove $\alpha_k \geq 0$ è il minimo della funzione monodimensionale

$$\phi_k(\alpha) = f\Big(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)\Big),$$

su tutti gli $\alpha \geq 0$. Quindi, per $\alpha \geq 0$, abbiamo

$$\phi_k(\alpha_k) \leq \phi_k(\alpha).$$

Applichiamo la regola della catena per calcolare la derivata prima nel punto $\alpha = 0$:

$$\phi_k'(0) = \frac{d}{d\alpha}\phi_k(0) = \nabla f\Big(\mathbf{x}_k - 0\nabla f(\mathbf{x}_k)\Big) \cdot \Big(-\nabla f(\mathbf{x}_k)\Big) = -\|\nabla f(\mathbf{x}_k)\|^2.$$

Per ipotesi $\nabla f(\mathbf{x}_k) \neq 0$, quindi

$$\phi_k'(0) = -\|\nabla f(x_k)\|^2 < 0.$$

Dato che abbiamo derivata prima negativa, ciò implica che esiste un $\bar{\alpha} > 0$ tale che $\phi_k(0) > \phi_k(\alpha)$ per ogni $\alpha \in (0, \bar{\alpha}]$. Perciò

$$f(\mathbf{x}_{k+1}) = \phi_k(\alpha_k) \leq \phi_k(\bar{\alpha}) < \phi_k(0) = f(\mathbf{x}_k).$$

□

In altre parole, ogni passo dell'algoritmo (tranne nei punti stazionari) produce una riduzione del valore della funzione obiettivo.

5.5 Metodi di Line Search

Come abbiamo appena visto, cercare il minimo α_k della funzione ϕ_k garantisce il massimo miglioramento possibile ad ogni passo, ma richiede la soluzione di un problema di ottimizzazione unidimensionale ad ogni iterazione.
Esistono vari metodi per trovare il valore ottimale, i cosiddetti Metodi di Line Search, che si dividono in due categorie principali:

- Exact Line search
- Inexact Line Search

5.5.1 Considerazioni sulla Scelta della Strategia di Line Search

La scelta del passo α_k gioca un ruolo fondamentale nell'efficacia dell'algoritmo. L'obiettivo della line search è determinare, ad ogni iterazione, il valore di α_k che minimizza la funzione monodimensionale

$$\phi_k(\alpha) = f\Big(\mathbf{x}_k - \alpha\nabla f(\mathbf{x}_k)\Big).$$

In teoria, il modo più diretto per trovare tale minimo consiste nel risolvere analiticamente l'equazione $\phi_k'(\alpha) = 0$. I metodi basati sul calcolo della derivata si ispirano a questo principio: utilizzano le informazioni del gradiente per individuare un passo ottimale che garantisca una riduzione efficace di f, spesso con un numero inferiore di iterazioni rispetto a strategie puramente empiriche.

Tuttavia, l'implementazione pratica di tali metodi può essere computazionalmente costosa, dato che le derivate spesso sono molto complicate da calcolare.

Per ovviare a queste limitazioni, i metodi di ricerca ad intervallo offrono un'alternativa robusta che richiede esclusivamente valutazioni della funzione obiettivo. Sebbene questi algoritmi garantiscano convergenza per classi ampie di funzioni e siano di implementazione relativamente semplice, essi convergono molto lentamente, e ciò li rende poco efficienti in contesti applicativi reali.

Per bilanciare accuratezza e costo computazionale, i metodi di Inexact Line Search, come quello di Armijo, offrono un compromesso efficace. Essi utilizzano informazioni del gradiente per garantire una discesa efficiente, ma accettano passi che, pur non essendo ottimali, assicurano una riduzione sufficiente della funzione obiettivo grazie a un meccanismo di backtracking.

5.6 Exact Line Search

L'exact line search trova il passo ottimale che minimizza esattamente la funzione lungo la direzione di discesa.
Cerchiamo quindi il valore α^* che soddisfa:

$$\phi'(\alpha^*) = 0.$$

5.6.1 Metodi basati sulla derivata

Metodo di Newton

Il metodo di Newton è una tecnica iterativa molto efficiente per la ricerca di minimi di una funzione. Supponiamo che per ogni punto α_k possiamo calcolare $\phi(\alpha_k)$, $\phi'(\alpha_k)$ e $\phi''(\alpha_k)$. Allora approssimiamo localmente la funzione $\phi(\alpha)$ con una parabola tangente al punto corrente, e ci spostiamo poi verso il minimo di tale parabola.

Derivazione della formula di Newton Sia $\phi(\alpha)$ una funzione di classe C^2 intervallo aperto centrato nel punto α_k. Possiamo approssimare ϕ localmente mediante il suo sviluppo di Taylor del secondo ordine attorno al punto corrente α_k:

$$q(\alpha) = \phi(\alpha_k) + \phi'(\alpha_k)(\alpha - \alpha_k) + \frac{1}{2}\phi''(\alpha_k)(\alpha - \alpha_k)^2.$$

Otteniamo così una parabola tangente a $\phi(\alpha_k)$, che approssima la curvatura locale. Il minimo di questa approssimazione quadratica si ottiene annullando la derivata prima:

$$0 = q'(\alpha) = \phi'(\alpha_k) + \phi''(\alpha_k)(\alpha - \alpha_k) \quad \iff \quad \alpha = \alpha_k - \frac{\phi'(\alpha_k)}{\phi''(\alpha_k)}.$$

Da cui ricaviamo la formula di approssimazione di Newton:

$$\alpha_{i+1} = \alpha_k - \frac{\phi'(\alpha_k)}{\phi''(\alpha_k)}.$$

Proprietà di convergenza Il metodo di Newton funziona bene se $\phi''(\alpha) > 0$ per ogni α. Intuitivamente, questo perché una derivata seconda positiva implica che la funzione è localmente convessa, e quindi l'approssimazione quadratica utilizzata dal metodo ha un minimo ben definito. In questo caso, il passo di Newton ci porta direttamente verso quel minimo. Il metodo converge molto rapidamente, ma richiede il calcolo della seconda derivata, che può risultare costoso o instabile in alcuni casi. Se invece $\phi''(\alpha) < 0$, il metodo di Newton può non convergere a un minimo (poiché l'approssimazione quadratica risulterebbe concava e il passo ci porterebbe verso un massimo invece che un minimo).

Costo computazionale Richiede il calcolo della derivata seconda, che può essere computazionalmente costoso o numericamente instabile.

Metodo della secante

Il metodo della secante è una variante del metodo di Newton, che evita di calcolare la derivata seconda, approssimandola tramite un rapporto incrementale.

Derivazione Approssimiamo la derivata seconda nella formula di Newton nel seguente modo:

$$\phi''(\alpha_k) \approx \frac{\phi'(\alpha_k) - \phi'(\alpha_{k-1})}{\alpha_k - \alpha_{k-1}}.$$

Sostituendo questa espressione nella formula di Newton si ottiene:

$$\alpha_{k+1} = \alpha_k - \phi'(\alpha_k)\frac{\alpha_k - \alpha_{k-1}}{\phi'(\alpha_k) - \phi'(\alpha_{k-1})}.$$

Geometricamente, questo corrisponde a trovare l'intersezione con l'asse delle ascisse della retta secante al grafico $\phi'(\alpha)$ passante per i punti $(\alpha_{k-1}, \phi'(\alpha_{k-1}))$ e $(\alpha_k, \phi'(\alpha_k))$.

Inizializzazione Per garantire convergenza, i punti iniziali α_0 e α_1 devono soddisfare la condizione

$$\phi'(\alpha_0)\phi'(\alpha_1) < 0,$$

assicurando così che esista un punto di minimo nell'intervallo $[\alpha_0, \alpha_1]$. Dopo la prima iterazione, usiamo semplicemente gli ultimi due punti correnti.

In pratica partiamo dal punto corrente $\alpha_0 = 0$. In questo punto abbiamo che

$$\phi_k'(0) = \nabla f\Big(\mathbf{x}_0 - 0\nabla f(\mathbf{x}_0)\Big) \cdot \Big(-\nabla f(\mathbf{x}_0)\Big) = -\|\nabla f(\mathbf{x}_0)\|^2 < 0,$$

quindi siamo in discesa.
Dobbiamo quindi trovare un secondo valore, il quale ha $\phi'(\alpha_1) > 0$.
Proviamo un secondo valore iniziale positivo come ad esempio $\alpha_1 = 1$ e valutiamo $\phi'(\alpha_1)$. Abbiamo due possibili risultati:

- se $\phi'(\alpha_1) > 0$, abbiamo un intervallo valido $[0, \alpha_1]$, e possiamo far partire l'algoritmo;
- se $\phi'(\alpha_1) < 0$, il minimo è più avanti, quindi raddoppiamo l'intervallo

$$\alpha_1 \to 2\alpha_1.$$

  Ripetiamo questa operazione finché $\phi'(\alpha_1) > 0$.

Osservazione 5.6.1. Può capitare che il valore per $\alpha_0 = 0$ è troppo negativo e quello per $\alpha_1 = 1$ è troppo positivo. Allora riduciamo α_1 finché otteniamo un valore positivo ragionevole, e facciamo la stessa cosa per α_0.

Poi ripetiamo questi passi finché il criterio di arresto

$$|\phi'(\alpha_{k+1})| < \varepsilon,$$

è soddisfatto.

Il metodo della secante converge più lentamente rispetto a Newton, ma è più pratico perché richiede solo le derivate prime. Per questo motivo è molto usato nelle ricerche lineari all'interno di algoritmi di ottimizzazione multidimensionale.

5.6.2 Metodi di ricerca ad intervalo

Metodo della sezione aurea

Il metodo della sezione aurea, o in inglese "Golden Section Search", è un metodo di ricerca monodimensionale usato per trovare il minimo α_* di una funzione continua $\phi : I \to \mathbb{R}$ in un intervallo chiuso $[a_0, b_0]$, senza usare derivate.

Definizione 5.6.2 (**Funzione unimodale**)**.** Una funzione $\phi : I \to \mathbb{R}$ si dice unimodale su un intervallo $[a_0, b_0]$ se esiste un unico punto α_* tale che

- ϕ è strettamente decrescente su $[a_0, \alpha^*]$;
- ϕ è strettamente crescente su $[\alpha^*, b]$.

Vediamo un esempio di una funzione unimodale nella Figura 5.1.

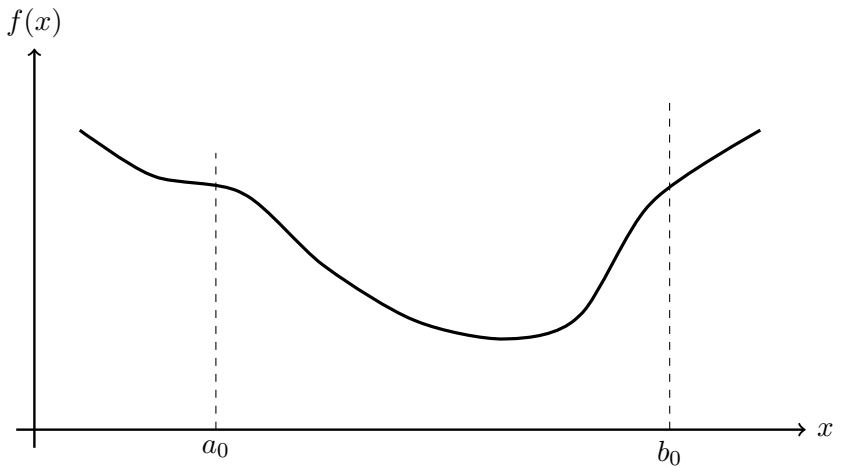


Figura 5.1: Funzione unimodale

Inizializzazione dell'intervallo L'idea è restringere man mano l'intervallo $[a_0, b_0]$ che contiene il minimo, confrontando valori della funzione in due punti interni.

Per fare ciò, è necessario determinare inizialmente un intervallo $[a_0, b_0]$, che contenga il minimo. Nel nostro caso specifico della funzione ϕ_k, tipicamente si fissa $a_0 = 0$, corrispondente al punto corrente $\mathbf{x}_k$ e si determina b_0 attraverso una procedura di espansione:

1. scegliamo $\alpha_0 = 0$ e un passo iniziale $\delta > 0$ (ad esempio $\delta = 1$);
2. calcoliamo
$$\phi(\delta) = f(\mathbf{x}_k - \delta \nabla f(\mathbf{x}_k));$$
3. confrontiamo con il valore nel punto corrente $\phi(0)$:
   - se
   $$\phi(\delta) > \phi(0),$$
   abbiamo superato il minimo, e quindi l'intervallo $[0, \delta]$ contiene il minimo;
   - se
   $$\phi(\delta) \leq \phi(0),$$
   siamo ancora in discesa, e quindi proseguiamo l'espansione raddoppiando δ e ripetiamo dal punto 2.

Alla fine della procedura, avremo trovato b_0 uguale al valore finale di δ tale che $\phi(b_0) > \phi(0)$, garantendo che l'intervallo $[0, b_0]$ contenga almeno un minimo locale.

Procedimento iterativo Definito il rapporto aureo $\rho = \frac{1+\sqrt{5}}{2} \approx 1.618$ e il suo reciproco $\tau = \frac{1}{\rho} = \rho - 1 \approx 0.618$, si procede iterativamente come segue.

Sia data una funzione $\phi : I \to \mathbb{R}$ unimodale in un intervallo iniziale $[a_0, b_0]$, ossia che l'intervallo $[a_0, b_0]$ contenga un minimo.

Ad ogni iterazione k, si calcolano due punti interni:

$$x_1 = b_k - \tau(b_k - a_k)$$
$$x_2 = a_k + \tau(b_k - a_k)$$

La simmetria di questa scelta è illustrata in Figura 5.2.

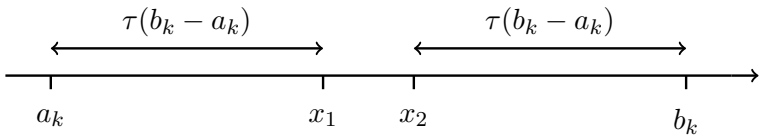


Figura 5.2: Disposizione simmetrica dei punti nel metodo della sezione aurea

Si valutano quindi i valori della funzione $\phi(x_1)$ e $\phi(x_2)$.

L'aggiornamento dell'intervallo avviene secondo la regola:

- se $\phi(x_1) > \phi(x_2)$: il minimo si trova in $[x_1, b_k]$, quindi
$$a_{k+1} = x_1, \quad b_{k+1} = b_k;$$
- se $\phi(x_1) < \phi(x_2)$: il minimo si trova in $[a_k, x_2]$, quindi
$$a_{k+1} = a_k, \quad b_{k+1} = x_2;$$
- se $\phi(x_1) = \phi(x_2)$: il minimo si trova in $[x_1, x_2]$, quindi
$$a_{k+1} = x_1, \quad b_{k+1} = x_2.$$

La Figura 5.3 illustra il caso $\phi_1(x_1) < \phi_2(x_2)$.

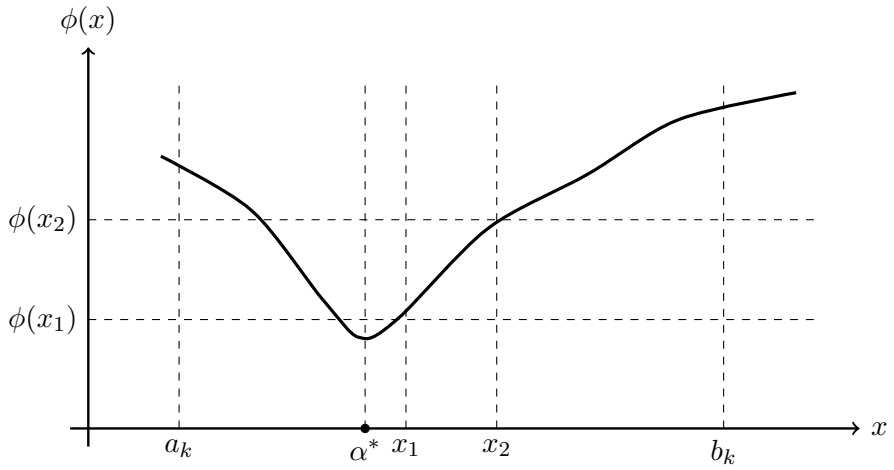


Figura 5.3: Caso $\phi(x_1) < \phi(x_2)$: il minimo è contenuto in $[a_k, x_2]$

Criterio di arresto e proprietà L'algoritmo termina quando la lunghezza dell'intervallo $[a_k, b_k]$ è inferiore a una tolleranza prefissata, ovvero, si sceglie un $\varepsilon > 0$, e si controlla che

$$|b_k - a_k| < \varepsilon.$$

Il metodo della sezione aurea riduce l'intervallo di incertezza di un fattore costante $\tau \approx 0.618$ ad ogni iterazione, garantendo convergenza lineare.

Metodo di Fibonacci

Anche il metodo di Fibonacci, come il metodo della sezione aurea, serve per trovare il minimo di una funzione unimodale su un intervallo $[a_0, b_0]$, senza usare la derivata prima. Invece di mantenere costante il rapporto di divisione, lo varia a ogni iterazione utilizzando i numeri di Fibonacci.

In parole povere, il metodo di Fibonacci è una versione più efficiente del metodo della sezione aurea che utilizza i numeri di Fibonacci per determinare i punti di valutazione.

Definizione 5.6.3 (**Successione di Fibonacci**)**.** La successione di Fibonacci è definita ricorsivamente come:

$$\begin{aligned} &F_0 = 0, \quad F_1 = 1 \\ &F_n = F_{n-1} + F_{n-2} \quad \text{per } n \geq 2 \end{aligned}$$

Di conseguenza, i suoi termini sono: $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$

Algoritmo Dato l'intervallo iniziale $[a_0, b_0]$ e il numero totale di iterazioni N, l'algoritmo procede come segue: per $k = 0, 1, \ldots, N-1$

1. calcoliamo i punti interni

$$\begin{aligned} x_1 &= a_k + \frac{F_{N-k-1}}{F_{N-k+1}}(b_k - a_k), \\ x_2 &= a_k + \frac{F_{N-k}}{F_{N-k+1}}(b_k - a_k); \end{aligned}$$

2. valutiamo $\phi(x_1)$ e $\phi(x_2)$;

3. aggiorniamo l'intervallo:

   - se $\phi(x_1) > \phi(x_2)$: $a_{k+1} = x_1$, $b_{k+1} = b_k$,
   - se $\phi(x_1) < \phi(x_2)$: $a_{k+1} = a_k$, $b_{k+1} = x_2$;

4. riutilizziamo il punto appropriato per l'iterazione successiva.

Iterazione finale e stima del minimo All'ultima iterazione ($k = N-1$) si verifica una situazione particolare: per $k = N-1$ abbiamo $F_{N-k} = F_1 = 1$ e $F_{N-k-1} = F_0 = 0$, quindi

$$\begin{aligned} x_1 &= a_{N-1} + \frac{F_0}{F_2}(b_{N-1} - a_{N-1}) = a_{N-1} \\ x_2 &= a_{N-1} + \frac{F_1}{F_2}(b_{N-1} - a_{N-1}) = a_{N-1} + \frac{1}{2}(b_{N-1} - a_{N-1}) \end{aligned}$$

I due punti x_1 e x_2 risultano molto vicini tra loro (uno coincide con a_{N-1}), rendendo impossibile determinare con precisione in quale sotto-intervallo si trovi il minimo. Pertanto, all'ultima iterazione si procede come segue:

- si valutano $\phi(x_1)$ e $\phi(x_2)$;
- si sceglie come stima del minimo il punto con il valore di funzione minore tra x_1 e x_2;
- alternativamente, per simmetria col metodo della sezione aurea, si può prendere la media: $\alpha^* = \frac{a_N + b_N}{2}$.

In pratica, la differenza tra le due scelte è trascurabile quando $[a_N, b_N]$ è sufficientemente piccolo: entrambe forniscono una stima accurata del punto di minimo entro la tolleranza prefissata.

Osservazione 5.6.4. Per $N \to \infty$, il rapporto tra numeri di Fibonacci consecutivi tende al rapporto aureo:

$$\lim_{n\to\infty} \frac{F_n}{F_{n+1}} = \frac{1}{\rho} \approx 0.618$$

Il metodo di Fibonacci risulta quindi asintoticamente equivalente al metodo della sezione aurea, ma è ottimale per un numero finito di valutazioni della funzione.

Continuiamo ora l'esempio precedente usando però il metodo della secante per il line search.

5.6.3 Esempio di Exact Line Search

Nel seguito verrà presentato un esempio dell'algoritmo di discesa più ripida con exact line search, realizzata mediante il metodo della secante. Successivamente, dopo aver introdotto l'Inexact Line Search, il medesimo problema verrà risolto utilizzando quest'ultima, in modo da poter confrontare le prestazioni dei due approcci.

Esempio 5.6.5. Usiamo il metodo della discesa più rapida per trovare il minimo della funzione

$$f(x_1, x_2, x_3) = (x_1 - 4)^4 + (x_2 - 3)^2 + 4(x_3 + 5)^4.$$

Partiamo dal punto $\mathbf{x}_0 = \Big(4, \quad 2, \quad -1\Big)$.
Come criterio di arrestamento scegliamo la condizione per la quale

$$\|\nabla f(\mathbf{x}_k)\| < 0.01.$$

Iterazione 1: Come spiegato nella Definizione 5.3.1, calcoliamo per prima cosa

$$\nabla f(x_1, x_2, x_3) = \Big(4(x_1 - 4)^3, \quad 2(x_2 - 3), \quad 16(x_3 + 5)^3\Big).$$

Nel punto iniziale abbiamo

$$\nabla f(4, 2, -1) = \Big(4(4 - 4)^3, \quad 2(2 - 3), \quad 16(-1 + 5)^3\Big) = \Big(0, \quad -2, \quad 1024\Big).$$

Calcoliamo il punto successivo dell'algoritmo applicando la formula della discesa del gradiente

$$\mathbf{x}_1 = \mathbf{x}_0 - \alpha \nabla f(\mathbf{x}_0).$$

Ma per calcolare $\mathbf{x}_1$, dobbiamo prima trovare il passo ottimale α_0 cioè il minimo della funzione

$$\phi(\alpha) = f(\mathbf{x}_0 - \alpha \nabla f(\mathbf{x}_0)) = (2\alpha - 1)^2 + 4(4 - 1024\alpha)^4.$$

La derivata di $\phi(\alpha)$ è quindi

$$\phi'(\alpha) = 4(2\alpha - 1) - 16384(4 - 1024\alpha)^3.$$

Per trovare gli zeri di $\phi'(\alpha)$ usiamo il metodo della secante.

Passo 1: Ricerca dei punti iniziali

Dobbiamo trovare due punti $\alpha_0.\alpha_1$ tali che

$$\phi'(\alpha_0)\phi'(\alpha_1) < 0.$$

Proviamo a scegliere $\alpha_0 = 0$ per cui otteniamo

$$\phi'(0) \approx -1.048 \times 10^6,$$

che è un valore negativo, però è molto grande e perciò il suo utilizzo prolungherebbe di molto il calcolo. Prima di risolvere tale problema cerchiamo un valore α_1 tale per cui $\phi(\alpha_1) > 0$. Partiamo con $\alpha_1 = 1$ e otteniamo

$$\phi'(1) = 4(2-1) - 16384(4-1024)^3 \approx 1.74 \times 10^{13},$$

che è maggiore di zero.

In questo caso abbiamo il problema che i valori assoluti sono troppo diversi (7 ordini di grandezza). Per velocizzare quindi la convergenza del metodo della secante aggiustiamo i valori α_0 e α_1 finché otteniamo dei valori ragionevoli.
Cominciamo riducendo α_1, fino ad ottenere un valore positivo ragionevolmente piccolo.

1. Supponiamo $\alpha_1 = 0.5$, quindi

$$\phi'(0.5) = 4(2 \times 0.5 - 1) - 16384(4 - 1024 \times 0.5)^3 \approx 2.15 \times 10^{12},$$

che è ancora troppo grande. Nella seguente tabella sono riportate le prove necessarie per trovare un valore adatto (i valori di ϕ' sono approssimati per mostrare l'ordine di grandezza):

Iterazione	α_1	$\phi'(\alpha_0)$
1	0.5	2.15×10^{12}
2	0.1	1.56×10^{10}
3	0.01	3.98×10^6
4	0.005	22.99×10^3
5	0.004	10.528

Per calcolare α_0 continuiamo a ridurre l'ultimo valore utile per α_1 e otteniamo

1. $\alpha_0 = 0.003$

$$\phi'(0.003) = 4(2 \times 0.003 - 1) - 16384(4 - 1024 \times 0.003)^3 = -13098.950,$$

che è un valore negativo, ragionevolmente piccolo.

Scegliamo quindi:

- $\alpha_0 = 0.003$,
- $\alpha_1 = 0.004$,
- $\phi'(0.003) = -13098.950$,
- $\phi'(0.004) = 10.528$.

Iterazione 1:

Iterazione 1.1
Applichiamo la formula:

$$\alpha_{k+1} = \alpha_k - \phi'(\alpha_k)\frac{\alpha_k - \alpha_{k-1}}{\phi'(\alpha_k) - \phi'(\alpha_{k-1})}$$

e otteniamo

$$\begin{aligned}\alpha_2 &= \alpha_1 - \phi'(\alpha_1)\frac{\alpha_1 - \alpha_0}{\phi'(\alpha_1) - \phi'(\alpha_0)} \\ &= 0.004 - 10.528 \times \frac{0.004 - 0.003}{10.528 - (-13098.950)} \\ &= 0.003999.\end{aligned}$$

Ora verifichiamo il criterio di arrestamento

$$|\phi'(\alpha_1)| < \varepsilon.$$

Imponiamo $\varepsilon = 0.1$, e calcoliamo

$$\phi'(\alpha_2) = 4(2\alpha_2 - 1) - 16384(4 - 1024\alpha_2)^3 = 10.16.$$

Il criterio di arrestamento quindi non è soddisfatto e proseguiamo con l'iterazione numero 2.
Iterazione 1.2: La formula ci dice

$$\alpha_3 = \alpha_2 - \phi'(\alpha_2)\frac{\alpha_2 - \alpha_1}{\phi'(\alpha_2) - \phi'(\alpha_1)},$$

dunque abbiamo:

- $\alpha_1 = 0.004$;
- $\alpha_2 = 0.003999$;
- $\phi'(\alpha_1) = 10.528$;
- $\phi'(\alpha_2) = 10.16$.

Otteniamo quindi

$$\begin{aligned}\alpha_3 &= 0.003999 - 10.16 \times \frac{0.003999 - 0.004}{10.16 - 10.528} \\ &= 0.003977.\end{aligned}$$

Calcoliamo

$$\phi'(\alpha_3) = 2.29,$$

e verifichiamo il criterio di arresto

$$|\phi'(\alpha_3)| > 0.1,$$

che non è soddisfatto e proseguiamo con l'iterazione numero 3.

Iterazione 1.3

$$\begin{aligned}\alpha_4 &= \alpha_3 - \phi'(\alpha_3)\frac{\alpha_3 - \alpha_2}{\phi'(\alpha_3) - \phi'(\alpha_2)} \\ &= 0.003977 - 2.29 \times \frac{0.003977 - 0.003999}{2.29 - 10.16} \\ &= 0.003971,\end{aligned}$$

Calcoliamo

$$\phi'(\alpha_4) = 0.74,$$

e verifichiamo il criterio di arresto

$$|\phi'(\alpha_4)| > 0.1,$$

che non è soddisfatto e proseguiamo con l'iterazione numero 4.

Iterazione 1.4

$$\begin{aligned}\alpha_5 &= \alpha_4 - \phi'(\alpha_4)\frac{\alpha_4 - \alpha_3}{\phi'(\alpha_4) - \phi'(\alpha_3)} \\ &= 0.003971 - 0.74 \times \frac{0.003971 - 0.003977}{0.74 - 2.29} \\ &= 0.003968,\end{aligned}$$

Calcoliamo

$$\phi'(\alpha_5) = 0.0999,$$

e verifichiamo il criterio di arresto

$$|\phi'(\alpha_5)| < 0.1.$$

Il criterio è soddisfatto.

Risultato della prima applicazione del metodo della secante

Dopo **4 iterazioni** del metodo della secante, otteniamo:

$$\alpha_0 = 0.003968.$$

Aggiornamento del punto

Possiamo ora calcolare il nuovo punto:

$$\begin{aligned}\mathbf{x}_1 &= \mathbf{x}_0 - \alpha_0 \nabla f(\mathbf{x}_0) \\ &= \begin{pmatrix}4, & 2, & -1\end{pmatrix} - 0.003968 \begin{pmatrix}0, & -2, & 1024\end{pmatrix} \\ &= \begin{pmatrix}4, & 2.007935, & -5.062853\end{pmatrix}.\end{aligned}$$

Iterazione 2:

Per trovare $\mathbf{x}_2$, determiniamo come prima cosa il gradiente in $\mathbf{x}_1$:

$$\begin{aligned}\nabla f(\mathbf{x}_1) &= \begin{pmatrix}4(x_1 - 4)^3, & 2(x_2 - 3), & 16(x_3 + 5)^3\end{pmatrix} \\ &= \begin{pmatrix}4(4 - 4)^3, & 2(2.007935 - 3), & 16(-5.062853 + 5)^3\end{pmatrix} \\ &= \begin{pmatrix}0, & -1.984, & -0.00397\end{pmatrix}.\end{aligned}$$

E cerchiamo α_1 che minimizza

$$\begin{aligned}\phi(\alpha) &= f(\mathbf{x}_1 - \alpha\nabla f(\mathbf{x}_1)) \\ &= (2,007935 + 1.984\alpha - 3)^2 + 4(-5.0628 + 0.00397\alpha + 5)^4 \\ &= (1.984\alpha - 0.9920)^2 + 4(-0.0628 + 0.00397\alpha)^4.\end{aligned}$$

Per fare ciò usiamo il metodo della secante, stavolta prendiamo i punti iniziali $\alpha_0 = 0$ e $\alpha_1 = 1$, in modo tale che:

$$\phi'(\alpha_0) < 0 \qquad \text{e} \qquad \phi'(\alpha_1) > 0.$$

Iterazione 2.1: La formula ci dice che

$$\alpha_2 = \alpha_1 - \phi'(\alpha_1)\frac{\alpha_1 - \alpha_0}{\phi'(\alpha_1) - \phi'(\alpha_0)},$$

e abbiamo:

- $\alpha_0 = 0$;
- $\phi'(\alpha_0) = -3.9368$;
- $\alpha_1 = 1$;
- $\phi(\alpha_1) = 3.9368$.

Otteniamo quindi

$$\begin{aligned}\alpha_3 &= 1 - 3.9368 \times \frac{1 - 0}{3.9368 - (-3.9368)} \\ &= 0.500002.\end{aligned}$$

Calcoliamo

$$\phi'(\alpha_2) = 5.471 \times 10^{-5},$$

e verifichiamo il criterio di arresto

$$|\phi'(\alpha_2)| < 0.1.$$

Il criterio è soddisfatto.

Aggiornamento del punto:

$$\begin{aligned}\mathbf{x}_2 &= \mathbf{x}_1 - \alpha_1\nabla f(\mathbf{x}_1) \\ &= \begin{pmatrix}4, & 3.00004, & -5.060867\end{pmatrix}\end{aligned}$$

Calcoliamo il gradiente finale:

$$\begin{aligned}\nabla f(\mathbf{x}_2) &= \begin{pmatrix}0, & 7.24 \times 10^{-6}, & -0.0036\end{pmatrix}, \\ \|\nabla f(\mathbf{x}_2)\| &= 1.301 \times 10^{-5} < 0.01.\end{aligned}$$

Abbiamo raggiunto la convergenza.

Risultato finale dopo 2 iterazioni:

Dopo 2 iterazioni del metodo del gradiente con ricerca lineare tramite metodo della secante, il gradiente è diventato molto piccolo, e possiamo quindi fermare le iterazioni, dato che ciò indica che siamo vicini a un punto stazionario. Infatti, il punto trovato è

$$\mathbf{x}^* \approx \Big(4, \quad 3.00004, \quad -5.060867\Big).$$

che è molto vicino al punto di minimo effettivo $\Big(4, \quad 3, \quad -5\Big)$.

5.7 Inexact Line Search

L'Inexact Line Search sacrifica l'ottimalità per l'efficienza computazionale. Scegliamo un learning rate in modo iterativo, riducendolo progressivamente finché la funzione $\phi(\alpha)$ diminuisce "a sufficienza".

5.7.1 Backtracking Line Search (Armijo Rule)

Questo metodo, noto anche come "backtracking line search" o "regola di Armijo", è ampiamente utilizzato per la sua semplicità e robustezza. Per poter applicare l'algoritmo dobbiamo scegliere due costanti $\beta \in (0,1)$ e $\gamma \in (0,1)$ fissate. Dato poi un punto corrente $\mathbf{x}_k$ e una direzione di discesa $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$, l'algoritmo procede come segue:

1. imponiamo $\alpha := 1$;
2. **while**
$$f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)) > f(\mathbf{x}_k) - \beta\alpha\|\nabla f(\mathbf{x}_k)\|^2,$$
3. **do**
$$\alpha := \gamma\alpha.$$

Una volta trovato il valore di α che soddisfa la condizione di arresto, esso viene definito come α_k, ovvero la lunghezza del passo utilizzata all'iterazione k per aggiornare il punto:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k).$$

5.7.2 Interpretazione della condizione

La condizione verifica che la riduzione della funzione sia almeno una frazione β della riduzione prevista dall'approssimazione lineare (cioè con Taylor).
Questa viene chiamata condizione di Armijo.

Definizione 5.7.1 (Condizione di Armijo)**.** Un passo α soddisfa la condizione di Armijo se

$$f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)) \leq f(\mathbf{x}_k) - \beta\alpha\|\nabla f(\mathbf{x}_k)\|^2 \tag{5.7.1}$$

Dato che $-\nabla f(\mathbf{x}_k)$ è una direzione di discesa, abbiamo $-\nabla f(\mathbf{x}_k) \cdot \nabla f(\mathbf{x}_k) = -\|\nabla f(\mathbf{x}_k)\|^2 < 0$, quindi il termine a destra rappresenta una decrescita garantita.

In questo algoritmo:

- β controlla la frazione minima di riduzione accettabile; in pratica assume valori tra $0.01 \leq \beta \leq 0.3$;
- γ controlla la velocità con cui il passo viene ridotto; tipicamente assume valori tra $0.1 \leq \gamma \leq 0.8$.

5.7.3 Interpretazione geometrica

Il metodo verifica che il valore della funzione $\phi(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{d}_k)$ sia inferiore a una stima lineare rilassata del valore atteso in base al gradiente. In altre parole, si accetta il passo solo se la funzione decresce in modo coerente con la direzione di discesa, garantendo stabilità e convergenza del metodo.

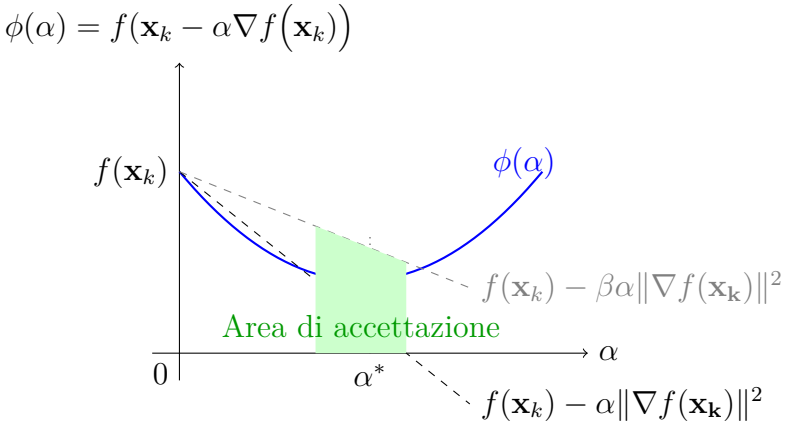


Figura 5.4: Rappresentazione del criterio di accettazione nel backtracking line search.

Nella Figura 5.4 è rappresentata la logica del criterio di accettazione del backtracking line search. La curva blu mostra l'andamento di $\phi(\alpha) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$, mentre la retta tratteggiata nera rappresenta la stima lineare del valore di f basata sull'espansione di Taylor al primo ordine:

$$L(\alpha) = f(\mathbf{x}_k) - \alpha \|\nabla f(\mathbf{x}_k)\|^2.$$

La linea grigia, con pendenza ridotta di un fattore β, definisce la condizione di accettazione del metodo, la condizione di Armijo:

$$A(\alpha) = f(\mathbf{x}_k) - \beta\alpha \|\nabla f(\mathbf{x}_k)\|^2.$$

Il passo α viene accettato solo se il valore di $f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$ risulta inferiore a questa retta, garantendo così una riduzione "sufficiente" del valore della funzione.

5.7.4 Esempio di Inexact Line Search

Come accennato in precedenza, vediamo ora un'implementazione dell'Inexact Line Search nell'algoritmo del steepest descent, per risolvere lo stesso problema.

Esempio 5.7.2. Usiamo il metodo della discesa più rapida per trovare il minimo della funzione

$$f(x_1, x_2, x_3) = (x_1 - 4)^4 + (x_2 - 3)^2 + 4(x_3 + 5)^4.$$

Partiamo dal punto $\mathbf{x}_0 = \begin{pmatrix} 4, & 2, & -1 \end{pmatrix}$.
Come criterio di arrestamento scegliamo la condizione per la quale

$$\|\nabla f(\mathbf{x}_k)\| < 0.01.$$

Come spiegato nella Definizione 5.3.1, calcoliamo per prima cosa

$$\nabla f(x_1, x_2, x_3) = \begin{pmatrix} 4(x_1 - 4)^3, & 2(x_2 - 3), & 16(x_3 + 5)^3 \end{pmatrix}.$$

Nel punto iniziale

$$\nabla f(4, 2, -1) = \begin{pmatrix} 4(4 - 4)^3, & 2(2 - 3), & 16(-1 + 5)^3 \end{pmatrix} = \begin{pmatrix} 0, & -2, & 1024 \end{pmatrix}.$$

Questa volta, per calcolare il valore di α usiamo l'Inexact Line Search.

Procedura di Inexact Line Search

Partendo con un valore iniziale $\alpha = 1$, dobbiamo verificare che la seguente condizione (5.7.1) viene soddisfatta, e in caso contrario lo cambiamo con

$$\alpha := \gamma\alpha.$$

Iterazione 1:

Riassumiamo i valori iniziali, scelti e calcolati in precedenza nella seguente tabella.

Parametro	Valore
$\mathbf{x}_0$	$\begin{pmatrix}4, & 2, & -1\end{pmatrix}$
$f(\mathbf{x}_0)$	1025
$\nabla f(\mathbf{x}_0)$	$\begin{pmatrix}0, & -2, & 1024\end{pmatrix}$
$\|\nabla f(\mathbf{x}_0)\|^2$	$1,048,580$
β	0.1
γ	0.5

Verifichiamo la condizione di Armijo, cominciamo con $\alpha = 1$:

$$\begin{aligned}
f(\mathbf{x}_0 - \alpha\nabla f(\mathbf{x}_0)) &\leq f(\mathbf{x}_0) - \beta\alpha\|\nabla f(\mathbf{x}_0)\|^2 \\
f\Big(\begin{pmatrix}4, & 2, & -1\end{pmatrix} - 1\begin{pmatrix}0, & -2, & 1025\end{pmatrix}\Big) &\leq 1025 - 0.1 \times 1 \times 1048580 \\
f\begin{pmatrix}4, & 4, & -1026\end{pmatrix} &\leq -103833 \\
(4-4)^4 + (4-3)^2 + 4(-1026+5)^4 &\leq -103833 \\
4.346 \times 10^{12} &\leq -103833.
\end{aligned}$$

Dunque, la condizione non è soddisfatta; quindi, cambiamo il valore di α:

$$\alpha := \gamma\alpha = 0.5 \times 1 = 0.5.$$

Ricalcoliamo.

Vediamo i successivi passi riassunti nella seguente tabella.

Tentativo	α	$f(\mathbf{x}_0 - \alpha\nabla f(\mathbf{x}_0))$	$f(\mathbf{x}_k) - \beta\alpha\|\nabla f(\mathbf{x}_k)\|^2$
1	1	4.33×10^{12}	$-103,833$
2	0.5	2.66×10^{11}	$-51,404$
3	0.25	1.61×10^{10}	$-25,189.5$
4	0.125	9.45×10^{8}	$-12,082.3$
5	0.0625	5.18×10^{7}	$-5,528.63$
6	0.03125	2.45×10^{6}	$-2,251.81$
7	0.015625	8.29×10^{5}	-613.40
8	0.0078125	1025	205.79
9	0.00390625	0.98	615.39

Risultato della ricerca lineare inesatta

Dopo 9 tentativi del metodo di Armijo si ottiene

$$\alpha_0 = 0.00390625.$$

Aggiornamento del punto

Possiamo ora calcolare il nuovo punto:

$$\begin{aligned}\mathbf{x}_1 &= \mathbf{x}_0 - \alpha_0 \nabla f(\mathbf{x}_0) \\ &= \begin{pmatrix}4, & 2, & -1\end{pmatrix} - 0.00390625 \times \begin{pmatrix}0, & -2, & 1024\end{pmatrix} \\ &= \begin{pmatrix}4, & 2.0078125, & -5\end{pmatrix}\end{aligned}$$

Verifichiamo il criterio di arresto:

$$\|\nabla f(\mathbf{x}_1)\| = 1.984375,$$

quindi non è ancora soddisfatto.

Iterazione 2:

Quindi applichiamo una seconda iterazione al punto $\mathbf{x}_1 = \begin{pmatrix}4, & 2.0078125, & -5\end{pmatrix}$. Calcoliamo i valori necessari.

Parametro	Valore
$\mathbf{x}_1$	$\begin{pmatrix}4, & 2.0078125, & -5\end{pmatrix}$
$f(\mathbf{x}_1)$	0.984436
$\nabla f(\mathbf{x}_1)$	$\begin{pmatrix}0, & -1.984375, & 0\end{pmatrix}$
$\|\nabla f(\mathbf{x}_1)\|^2$	3.937744
β	0.1
γ	0.5

Verifichiamo la condizione di Armijo, cominciamo con $\alpha = 1$:

$$\begin{aligned}f(\mathbf{x}_0 - \alpha \nabla f(\mathbf{x}_0)) &\leq f(\mathbf{x}_0) - \beta\alpha\|\nabla f(\mathbf{x}_0)\|^2 \\ (0)^4 + (3.937744 - 3)^2 + 4(0)^4 &\leq 0.984436 - 0.1 \times 3.937744 \\ 0.984436 &\leq 0.590662.\end{aligned}$$

La condizioni quindi non è ancora soddisfatta, e prendiamo il valore aggiornato $\alpha = 0.5$:

$$\begin{aligned}\mathbf{x}_1 - \alpha \nabla f(\mathbf{x}_1) &= \begin{pmatrix}4, & \begin{pmatrix}2.0078125 - 0.5 \times (-1.984375)\end{pmatrix}, & -5\end{pmatrix} = \begin{pmatrix}4, & 3, & -5\end{pmatrix}, \\ f(\mathbf{x}_1 - \alpha \nabla f(\mathbf{x}_1)) &= (0)^4 + (3-3)^2 + 4(0)^4 = 0, \\ f(\mathbf{x}_1) - \beta\alpha\|\nabla f(\mathbf{x}_1)\|^2 &= 0.984436 - 0.1 \times 0.5 \times 3.937744 = 0.787549.\end{aligned}$$

Quindi

$$0 \leq 0.787549$$

che è vero. Perciò la condizione è soddisfatta.

Riassumiamo il procedimento con la seguente tabella.

Tentativo	α	$f(\mathbf{x}_0 - \alpha \nabla f(\mathbf{x}_0))$	$f(\mathbf{x}_k) - \beta\alpha\|\nabla f(\mathbf{x}_k)\|^2$
1	1	0.98	0.59
2	0.5	0	0.87

Risultato della ricerca lineare inesatta

Dopo 2 tentativi del metodo di Armijo si ottiene

$$\alpha_1 = 0.5.$$

Aggiornamento del punto

Possiamo ora calcolare il nuovo punto:

$$\begin{aligned}\mathbf{x}_2 &= \mathbf{x}_1 - \alpha_1 \nabla f(\mathbf{x}_1) \\ &= \begin{pmatrix} 4, & 2.0078125, & -5 \end{pmatrix} - 0.5 \begin{pmatrix} 0, & -1.984375, & 0 \end{pmatrix} \\ &= \begin{pmatrix} 4, & 3, & -5 \end{pmatrix}.\end{aligned}$$

Verifichiamo il criterio di arresto:

$$\nabla f(\mathbf{x}_2) = \begin{pmatrix} 0, & 0, & 0 \end{pmatrix},$$
$$\|\nabla f(\mathbf{x}_2)\| = 0 \leq 0.01.$$

In effetti, abbiamo trovato il punto minimo esatto, visto che il gradiente si annulla.

Osservazione 5.7.3. Nel metodo della discesa più rapida, l'Inexact Line Search presenta diversi vantaggi rispetto all'exact line search. Mentre la ricerca esatta del passo ottimale richiede la minimizzazione completa della funzione lungo la direzione di discesa — operazione spesso costosa o impraticabile — la ricerca approssimata riduce notevolmente il costo computazionale, consentendo di determinare un passo "sufficientemente buono" con poche valutazioni della funzione e del gradiente. Inoltre, l'Inexact Line Search mantiene una buona velocità di convergenza teorica e pratica, evitando oscillazioni o passi eccessivamente piccoli che possono derivare da un calcolo troppo preciso. In sintesi, offre un compromesso efficiente tra accuratezza e costo computazionale, rendendo il metodo più rapido e stabile nelle applicazioni reali.

5.8 Analisi della convergenza per funzioni di costo quadratiche

5.8.1 La Funzione di Lyapunov

Per rendere l'analisi della convergenza più conveniente, al posto di lavorare con f, definiamo $V(\mathbf{x})$ come la misura d'errore rispetto al minimo $\mathbf{x}^*$:

$$V(\mathbf{x}) = f(\mathbf{x}) - f(\mathbf{x}^*).$$

Questa funzione non è un'altra funzione da minimizzare, ma serve come funzione di Lyapunov, cioè come misura dell'errore del sistema.
In altre parole $V(\mathbf{x})$ misura quanto siamo lontani dal minimo di $\mathbf{x}^*$.

Nel caso più trattato nell'analisi del gradiente, quello quadratico, abbiamo

$$V(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\top \mathbf{Q}(\mathbf{x} - \mathbf{x}^*),$$

dove

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{b}^\top \mathbf{x} + c,$$

con minimo in

$$\mathbf{x}^* = \mathbf{Q}^{-1}\mathbf{b}.$$

Come ricavare V(x)

Caso generale: Partiamo da una funzione qualsiasi $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$ che vogliamo minimizzare, e sia $\mathbf{x}^*$ un punto di minimo globale. Allora il modo naturale per calcolare quanto siamo lontani dall'ottimo di f è fare la differenza fra i valori di f e il suo minimo:

$$V(\mathbf{x}) = f(\mathbf{x}) - f(\mathbf{x}^*).$$

In particolare, dato che $\mathbf{x}^*$ è un minimo, abbiamo $V(\mathbf{x}) \geq 0$ e assume il valore 0 nel caso in cui $\mathbf{x} = \mathbf{x}^*$.

Osservazione 5.8.1. Le informazioni locali su f si trasferiscono su V, cioè

$$\nabla V(\mathbf{x}) = \nabla f(\mathbf{x}),$$
$$\mathbf{H}V(\mathbf{x}) = \mathbf{H}f(\mathbf{x}).$$

Caso quadratico: Assumiamo che f sia quadratica nella forma canonica

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{b}^\top \mathbf{x} + c,$$

con $\mathbf{Q} = \mathbf{Q}^\top \succ 0$, per il Teorema 1.6.15 possiamo quindi calcolare l'inversa di $\mathbf{Q}$.
Allora il minimo $\mathbf{x}^*$ è dato risolvendo

$$\nabla f(\mathbf{x}) = \mathbf{Q}\mathbf{x} - \mathbf{b} = 0,$$

cioè:

$$\mathbf{x}^* = \mathbf{Q}^{-1}\mathbf{b}.$$

Ora calcoliamo $f(\mathbf{x}) - f(\mathbf{x}^*)$.
Abbiamo

$$f(\mathbf{x}) - f(\mathbf{x}^*) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \frac{1}{2}(\mathbf{x}^*)^\top \mathbf{Q}\mathbf{x}^* - \mathbf{b}^\top(\mathbf{x} - \mathbf{x}^*),$$

a questo punto, usiamo $\mathbf{b} = \mathbf{Q}\mathbf{x}^*$ e applichiamo la proprietà di trasposizione nei prodotti scalari, per ottenere

$$\begin{aligned} f(\mathbf{x}) - f(\mathbf{x}^*) &= \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \frac{1}{2}(\mathbf{x}^*)^\top \mathbf{Q}\mathbf{x}^* - (\mathbf{x} - \mathbf{x}^*)^\top \mathbf{Q}\mathbf{x}^*, \\ &= \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{x}^\top \mathbf{Q}\mathbf{x}^* + \frac{1}{2}(\mathbf{x}^*)^\top \mathbf{Q}\mathbf{x}^*, \\ &= \frac{1}{2}\Big(\mathbf{x}^\top \mathbf{Q}\mathbf{x} - 2\mathbf{x}^\top \mathbf{Q}\mathbf{x}^* + (\mathbf{x}^*)^\top \mathbf{Q}\mathbf{x}^*\Big), \\ &= \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\top \mathbf{Q}(\mathbf{x} - \mathbf{x}^*), \\ &= V(\mathbf{x}). \end{aligned}$$

Osservazione 5.8.2. Il vantaggio di questa forma è che è chiaramente positiva e vale zero se e solo se $\mathbf{x} - \mathbf{x}^* = 0$. Quindi è perfetta per misurare la distanza dal minimo.

Per analizzare la convergenza del metodo del gradiente per il problema quadratico, risulta utile studiare l'evoluzione della funzione di Lyapunov $V(\mathbf{x})$.

Il seguente lemma fornisce una relazione fondamentale che lega il valore di V in due iterazioni successive, quantificando esattamente la riduzione dell'errore in un singolo passo del metodo.

Notazione 5.8.3. Per semplicità, da questo punto in avanti poniamo

$$\mathbf{g}_k := \nabla f(\mathbf{x}_k).$$

Lemma 5.8.4. *Sia*

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{b}^\top \mathbf{x} + c$$

una funzione quadratica, con $\mathbf{Q} \in \mathbb{R}^{n\times n}$ *simmetrica e definita positiva (*$\mathbf{Q} = \mathbf{Q}^\top \succ 0$*), e sia* $\mathbf{x}^* = \mathbf{Q}^{-1}\mathbf{b}$ *il suo unico punto di minimo.*

Sia $\{\mathbf{x}_k\}$ *la successione generata dal metodo del gradiente:*

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k), \tag{5.8.1}$$

dove $\nabla f(\mathbf{x}_k) = \mathbf{Q}\mathbf{x}_k - \mathbf{b}$ *e* $\alpha_k > 0$ *è il passo di discesa.*

Allora per ogni $k \geq 0$ *abbiamo*

$$V(\mathbf{x}_{k+1}) = (1 - \gamma_k)V(\mathbf{x}_k), \tag{5.8.2}$$

dove il coefficiente di riduzione γ_k *è dato da:*

$$\gamma_k = \begin{cases} 1 & \text{se } \mathbf{g}_k = \mathbf{0}, \\ \dfrac{\alpha_k \mathbf{g}_k^\top \mathbf{g}_k - \frac{1}{2}\alpha_k^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}{V(\mathbf{x}_k)} & \text{se } \mathbf{g}_k \neq \mathbf{0}. \end{cases}$$

Inoltre, se $\mathbf{g}_k \neq \mathbf{0}$ *e* α_k *è scelto in modo che*

$$0 < \alpha_k < \frac{2\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k},$$

allora $0 < \gamma_k < 1$.

Dimostrazione. La dimostrazione procede per calcolo diretto. Si noti innanzitutto, che, se $\mathbf{g}_k = \nabla f(\mathbf{x}_k) = \mathbf{0}$, allora $\mathbf{x}_k = \mathbf{x}^*$ e per definizione $V(\mathbf{x}_{k+1}) = 0$. Poiché anche $V(\mathbf{x}_k) = 0$, la relazione (5.8.2) è verificata per qualsiasi γ_k, e scegliamo $\gamma_k = 1$.

Per $\mathbf{g}_k \neq 0$, grazie all'equazione (5.8.2), vediamo che

$$\gamma_k = \frac{V(\mathbf{x}_k) - V(\mathbf{x}_{k+1})}{V(\mathbf{x}_k)}. \tag{5.8.3}$$

Per facilitare i calcoli poniamo $\mathbf{y}_k = \mathbf{x}_k - \mathbf{x}^*$. Allora

$$V(\mathbf{x}_k) = \frac{1}{2}\mathbf{y}_k^\top \mathbf{Q}\mathbf{y}_k.$$

Grazie alla formula (5.8.1), otteniamo

$$\begin{aligned} \mathbf{y}_{k+1} &= \mathbf{x}_{k+1} - \mathbf{x}^* \\ &= \mathbf{x}_k - \alpha_k \mathbf{g}_k - \mathbf{x}^* \\ &= \mathbf{y}_k - \alpha_k \mathbf{g}_k, \end{aligned}$$

dove

$$\mathbf{g}_k = \nabla f(\mathbf{x}_k) = \mathbf{Q}\mathbf{x}_k - \mathbf{b}.$$

Sostituendo $\mathbf{b} = \mathbf{Q}\mathbf{x}^*$, otteniamo

$$\mathbf{g}_k = \nabla f(\mathbf{x}_k) = \mathbf{Q}(\mathbf{x}_k - \mathbf{x}^*) = \mathbf{Q}\mathbf{y}_k.$$

Inseriamo questa identità nell'equazione precedente:

$$\begin{aligned}\mathbf{y}_{k+1} &= \mathbf{y}_k - \alpha_k \mathbf{Q}\mathbf{y}_k \\ &= (\mathbf{I} - \alpha_k \mathbf{Q})\mathbf{y}_k.\end{aligned}$$

Usando la definizione di V possiamo esprimere $V(\mathbf{x}_{k+1})$ come

$$\begin{aligned}V(\mathbf{x}_{k+1}) &= \frac{1}{2}\mathbf{y}_{k+1}^\top \mathbf{Q}\mathbf{y}_{k+1} \\ &= \frac{1}{2}(\mathbf{y}_k)^\top (\mathbf{I} - \alpha_k \mathbf{Q})^\top \mathbf{Q}(\mathbf{I} - \alpha_k \mathbf{Q})\mathbf{y}_k.\end{aligned}$$

Sviluppando il prodotto centrale, troviamo

$$(\mathbf{I} - \alpha_k \mathbf{Q})^\top \mathbf{Q}(\mathbf{I} - \alpha_k \mathbf{Q}) = (\mathbf{I}^\top - \alpha_k \mathbf{Q}^\top)(\mathbf{Q} - \alpha_k \mathbf{Q}^2).$$

Dato che la matrice $\mathbf{Q}$ è simmetrica, allora $(\mathbf{I} - \alpha_k \mathbf{Q})^\top = \mathbf{I} - \alpha_k \mathbf{Q}$ e quindi abbiamo

$$\begin{aligned}(\mathbf{I} - \alpha_k \mathbf{Q})^\top (\mathbf{Q} - \alpha_k \mathbf{Q}^2) &= (\mathbf{I} - \alpha_k \mathbf{Q})(\mathbf{Q} - \alpha_k \mathbf{Q}^2) \\ &= \mathbf{Q} - \alpha_k \mathbf{Q}\mathbf{Q} - \alpha_k \mathbf{Q}^2 + \alpha_k^2 \mathbf{Q}\mathbf{Q}^2 \\ &= \mathbf{Q} - 2\alpha_k \mathbf{Q}^2 + \alpha_k^2 \mathbf{Q}^3,\end{aligned}$$

da cui

$$V(\mathbf{x}_{k+1}) = \frac{1}{2}\mathbf{y}_k^\top \Big(\mathbf{Q} - 2\alpha_k \mathbf{Q}^2 + \alpha_k^2 \mathbf{Q}^3\Big)\mathbf{y}_k.$$

Riscriviamo i seguenti termini in funzione di $\mathbf{g}_k = \mathbf{Q}\mathbf{y}_k$, sfruttando sempre la simmetria di $\mathbf{Q}$:

$$\begin{aligned}(\mathbf{y}_k)^\top \mathbf{Q}^2 \mathbf{y}_k &= (\mathbf{Q}\mathbf{y}_k)^\top (\mathbf{Q}\mathbf{y}_k) = \mathbf{g}_k^\top \mathbf{g}_k, \\ (\mathbf{y}_k)^\top \mathbf{Q}^3 \mathbf{y}_k &= (\mathbf{Q}\mathbf{y}_k)^\top \mathbf{Q}(\mathbf{Q}\mathbf{y}_k) = \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k.\end{aligned}$$

Sostituendo questi risultati nell'equazione precedente otteniamo l'identità fondamentale

$$\begin{aligned}V(\mathbf{x}_{k+1}) &= \frac{1}{2}(\mathbf{y}_k)^\top \mathbf{Q}\mathbf{y}_k - \alpha_k (\mathbf{y}_k)^\top \mathbf{Q}^2 \mathbf{y}_k + \frac{1}{2}(\mathbf{y}_k)^\top \alpha_k^2 \mathbf{Q}^3 \mathbf{y}_k \\ &= V(\mathbf{x}_k) - \alpha_k \mathbf{g}_k^\top \mathbf{g}_k + \frac{1}{2}\alpha_k^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k. \qquad (5.8.4)\end{aligned}$$

Ora, per determinare un'espressione esplicita del coefficiente di riduzione γ_k, rimaneggiamo l'equazione (5.8.4):

$$V(\mathbf{x}_k) - V(\mathbf{x}_{k+1}) = \alpha_k \mathbf{g}_k^\top \mathbf{g}_k - \frac{1}{2}\alpha_k^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k.$$

Dividendo entrambi membri per $V(\mathbf{x}_k)$, e sfruttando l'equazione (5.8.3), otteniamo

$$\gamma_k = \frac{\alpha_k \mathbf{g}_k^\top \mathbf{g}_k - \frac{1}{2}\alpha_k^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}{V(\mathbf{x}_k)}.$$

Dimostrazione che $0 < \gamma_k < 1$: Per dimostrare che $\gamma_k > 0$, osserviamo che:

$$\alpha_k \mathbf{g}_k^\top \mathbf{g}_k - \frac{1}{2}\alpha_k^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k > 0$$

se e solo se

$$\alpha_k \mathbf{g}_k^\top \mathbf{g}_k > \frac{1}{2}\alpha_k^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k.$$

Dividendo per $\alpha_k > 0$

$$\mathbf{g}_k^\top \mathbf{g}_k > \frac{1}{2}\alpha_k \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k,$$

da cui

$$\alpha_k < \frac{2\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}.$$

Questa condizione è verificata per ipotesi, quindi $\gamma_k > 0$.

Per dimostrare che $\gamma_k < 1$, osserviamo che dalla (5.8.4)

$$V(\mathbf{x}_{k+1}) = V(\mathbf{x}_k) - \left(\alpha_k \mathbf{g}_k^\top \mathbf{g}_k - \frac{1}{2}\alpha_k^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k\right).$$

Poiché la quantità tra parentesi quadre è positiva (come appena dimostrato), abbiamo $V(\mathbf{x}_{k+1}) < V(\mathbf{x}_k)$, e quindi

$$\gamma_k = \frac{V(\mathbf{x}_k) - V(\mathbf{x}_{k+1})}{V(\mathbf{x}_k)} < 1.$$

□

Il Lemma 5.8.4 fornisce una relazione ricorsiva per l'evoluzione dell'errore: ad ogni passo l'energia residua $V(\mathbf{x}_k)$ viene moltiplicata per un fattore $(1-\gamma_k) < 1$, fornendo così la base per lo studio della convergenza dell'algoritmo.

Osservazione 5.8.5. Poiché $V(\mathbf{x}_{k+1}) \geq 0$, si può osservare che $1 - \gamma_k \geq 0$, e quindi $\gamma_k \leq 1$, anche senza bisogno di ipotesi aggiuntive su α_k.
Se inoltre $\gamma_k > 0$, allora $V(\mathbf{x}_{k+1}) < V(\mathbf{x}_k)$ e abbiamo anche qua la proprietà di decrescita.

Osservazione 5.8.6. Possiamo riscrivere γ_k nel seguente modo. Tenendo conto che

$$V(\mathbf{x}_{k+1}) = V(\mathbf{x}_k) - \alpha_k \mathbf{g}_k^\top \mathbf{g}_k + \frac{1}{2}\alpha_k^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k,$$
$$V(\mathbf{x}_k) = \frac{1}{2}\mathbf{y}_k^\top \mathbf{Q}\mathbf{y}_k,$$

e inserendoli nella formula per γ_k nel caso $V(x_k) \neq 0$, otteniamo

$$\begin{aligned}\gamma_k &= \frac{V(\mathbf{x}_k) - V(\mathbf{x}_{k+1})}{V(\mathbf{x}_k)} \\ &= \frac{V(\mathbf{x}_k) - (V(\mathbf{x}_k) - \alpha_k \mathbf{g}_k^\top \mathbf{g}_k - \frac{1}{2}\alpha_k^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k)}{\frac{1}{2}\mathbf{y}_k^\top \mathbf{Q}\mathbf{y}_k} \\ &= \frac{2\alpha_k \mathbf{g}_k^\top \mathbf{g}_k + \alpha_k^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}{\mathbf{y}_k^\top \mathbf{Q}\mathbf{y}_k}\end{aligned}$$

Ricordiamo che

$$\mathbf{g}_k = \mathbf{Q}\mathbf{y}_k,$$

da qui segue

$$\mathbf{y}_k = \mathbf{Q}^{-1}\mathbf{g}_k.$$

Il denominatore diventa quindi

$$\mathbf{y}_k^\top \mathbf{Q}\mathbf{y}_k = (\mathbf{Q}^{-1}\mathbf{g}_k)^\top \mathbf{Q}(\mathbf{Q}^{-1}\mathbf{g}_k) = \mathbf{g}_k^\top \mathbf{Q}^{-1}\mathbf{g}_k.$$

Otteniamo perciò

$$\gamma_k = \frac{2\alpha_k \mathbf{g}_k^\top \mathbf{g}_k + \alpha_k^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}^{-1}\mathbf{g}_k}.$$

Mettiamo in evidenza $\alpha_k \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}^{-1}\mathbf{g}_k}$, cioè dividiamo e moltiplichiamo opportunamente il numeratore

$$2\alpha_k \mathbf{g}_k^\top \mathbf{g}_k + \alpha_k^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k = \alpha_k \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k \Big(2\frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k} - \alpha_k \Big),$$

e otteniamo

$$\gamma_k = \alpha_k \frac{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}^{-1}\mathbf{g}_k} \Big(2\frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k} - \alpha_k \Big). \tag{5.8.5}$$

5.8.2 Teorema di convergenza globale

Siamo ora pronti per dimostrare il risultato principale di questa sezione: la condizione necessaria e sufficiente affinché il metodo del gradiente converga globalmente al minimo, nel caso della funzione quadratica.

La disuguaglianza di Rayleigh è una relazione che collega gli autovalori di una matrice simmetrica definita positiva con il rapporto tra un vettore non nullo $\mathbf{x} \in \mathbb{R}^n$ e la matrice stessa.

Proposizione 5.8.7 (**Disuguaglianza di Rayleigh**). *Sia $\mathbf{Q} \in \mathbb{R}^{n\times n}$ una matrice simmetrica definita positiva. Allora per ogni vettore $\mathbf{x} \in \mathbb{R}^n$ vale*

$$\lambda_{\min}(\mathbf{Q})\|\mathbf{x}\|^2 \leq \mathbf{x}^\top \mathbf{Q}\mathbf{x} \leq \lambda_{\max}(\mathbf{Q})\|\mathbf{x}\|^2, \tag{5.8.6}$$

dove

- $\lambda_{\min}(\mathbf{Q})$ *denota il minimo autovalore di* $\mathbf{Q}$,
- $\lambda_{\max}(\mathbf{Q})$ *denota il massimo autovalore di* $\mathbf{Q}$.

Inoltre, valgono le relazioni

$$\lambda_{\min}(\mathbf{Q}^{-1}) = \frac{1}{\lambda_{\max}(\mathbf{Q})}$$
$$\lambda_{\max}(\mathbf{Q}^{-1}) = \frac{1}{\lambda_{\min}(\mathbf{Q})}$$

e quindi

$$\frac{1}{\lambda_{\max}(\mathbf{Q})}\|\mathbf{x}\|^2 \leq \mathbf{x}^\top \mathbf{Q}^{-1}\mathbf{x} \leq \frac{1}{\lambda_{\min}(\mathbf{Q})}\|\mathbf{x}\|^2.$$

Dimostrazione. La dimostrazione si basa sul fatto che, quando abbiamo una matrice simmetrica, possiamo sempre "ruotare" il sistema di coordinate per allinearlo con le direzioni principali della matrice, cioè i suoi autovettori.

Per il Teorema Spettrale 1.6.9, ogni matrice simmetrica $\mathbf{Q}$ si può scrivere come

$$\mathbf{Q} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top,$$

dove $\mathbf{\Lambda}$ è una matrice diagonale contenente gli autovalori

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} = \operatorname{diag}(\lambda_1, \dots, \lambda_n),$$

e $\mathbf{V}$ una matrice ortogonale ($\mathbf{V}^\top\mathbf{V} = \mathbf{I}$) le cui colonne sono gli autovettori di $\mathbf{Q}$.

Prendiamo un qualsiasi vettore $\mathbf{x}$ e definiamo

$$\mathbf{y} = \mathbf{V}^\top\mathbf{x},$$

ovvero, esprimiamo $\mathbf{x}$ nelle coordinate degli autovettori di $\mathbf{Q}$.
Poiché $\mathbf{V}$ è ortogonale, per il Lemma 1.4.29 la lunghezza del vettore non cambia:

$$\|\mathbf{y}\| = \|\mathbf{V}^\top\mathbf{x}\| = \|\mathbf{x}\|.$$

Sostituiamo ora la forma diagonalizzata di $\mathbf{Q}$

$$\mathbf{x}^\top\mathbf{Q}\mathbf{x} = \mathbf{x}^\top(\mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top)\mathbf{x},$$

e raggruppiamo i termini

$$(\mathbf{V}^\top\mathbf{x})^\top\mathbf{\Lambda}(\mathbf{V}^\top\mathbf{x}) = \mathbf{y}^\top\mathbf{\Lambda}\mathbf{y}.$$

Ma $\mathbf{\Lambda}$ è diagonale, quindi per l'Osservazione 1.4.7 abbiamo

$$\mathbf{x}^\top\mathbf{Q}\mathbf{x} = \mathbf{y}^\top\mathbf{\Lambda}\mathbf{y} = \sum_{i=1}^{n}\lambda_i y_i^2 = \lambda_1 y_1^2 + \lambda_2 y_2^2 + \cdots + \lambda_n y_n^2.$$

Ora, sia $\lambda_{\min}$ il più piccolo autovalore e $\lambda_{\max}$ il più grande.
Allora abbiamo che per il limite inferiore

$$\begin{aligned}\lambda_1 y_1^2 + \lambda_2 y_2^2 + \cdots + \lambda_n y_n^2 &\geq \lambda_{\min} y_1^2 + \lambda_{\min} y_2^2 + \cdots + \lambda_{\min} y_n^2 \\ &= \lambda_{\min}(y_1^2 + y_2^2 + \cdots + y_n^2) \\ &= \lambda_{\min}\|\mathbf{y}\|^2 = \lambda_{\min}\|\mathbf{x}\|^2.\end{aligned}$$

E per il limite superiore

$$\begin{aligned}\lambda_1 y_1^2 + \lambda_2 y_2^2 + \cdots + \lambda_n y_n^2 &\leq \lambda_{\max} y_1^2 + \lambda_{\max} y_2^2 + \cdots + \lambda_{\max} y_n^2 \\ &= \lambda_{\max}(y_1^2 + y_2^2 + \cdots + y_n^2) \\ &= \lambda_{\max}\|\mathbf{y}\|^2 = \lambda_{\max}\|\mathbf{x}\|^2.\end{aligned}$$

Mettendo tutto insieme otteniamo

$$\lambda_{\min}\|\mathbf{x}\|^2 \leq \mathbf{x}^\top\mathbf{Q}\mathbf{x} \leq \lambda_{\max}\|\mathbf{x}\|^2.$$

Poiché $\mathbf{Q}$ è definita positiva, tutti i suoi autovalori sono strettamente positivi

$$\lambda_i(\mathbf{Q}) > 0 \qquad \text{per ogni} \qquad i = 1, \ldots, n.$$

Di conseguenza, per il Teorema 1.6.15 $\mathbf{Q}$ è invertibile.
Inoltre per il Teorema 1.6.16 possiamo scrivere tale inversa come

$$\mathbf{Q}^{-1} = \mathbf{V}\mathbf{\Lambda}^{-1}\mathbf{V}^\top,$$

dove

$$\mathbf{\Lambda}^{-1} = \operatorname{diag}\left(\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \ldots, \frac{1}{\lambda_n}\right).$$

Pertanto, gli autovalori di $\mathbf{Q}^{-1}$ sono gli inversi dei autovalori di $\mathbf{Q}$

$$\lambda_i(\mathbf{Q}^{-1}) = \frac{1}{\lambda_i(\mathbf{Q})}.$$

Applicando la disuguaglianza di Rayleigh dimostrata in precedenza alla matrice $\mathbf{Q^{-1}}$, otteniamo

$$\lambda_{\min}(\mathbf{Q}^{-1})\|\mathbf{x}\|^2 \leq \mathbf{x}^\top \mathbf{Q}^{-1}\mathbf{x} \leq \lambda_{\max}(\mathbf{Q}^{-1})\|\mathbf{x}\|^2,$$

cioè

$$\frac{1}{\lambda_{\max}(\mathbf{Q})}\|\mathbf{x}\|^2 \leq \mathbf{x}^\top \mathbf{Q}^{-1}\mathbf{x} \leq \frac{1}{\lambda_{\min}(\mathbf{Q})}\|\mathbf{x}\|^2.$$

□

La forma quadratica $\mathbf{x}^\top\mathbf{Q}\mathbf{x}$ può essere interpretata come una "lunghezza" distorta di $\mathbf{x}$, dove la distorsione è determinata dagli autovalori di $\mathbf{Q}$ lungo le direzioni dei suoi autovettori.

La disuguaglianza di Rayleigh ci dice che il rapporto $\frac{\mathbf{x}^\top\mathbf{Q}\mathbf{x}}{\|\mathbf{x}\|^2}$ sarà sempre compreso tra il valore minimo e il massimo degli autovalori di $\mathbf{Q}$.

Osservazione 5.8.8. Se $\mathbf{x}$ è un autovettore di $\mathbf{Q}$ corrispondente all'autovalore λ, allora

$$\mathbf{x}^\top\mathbf{Q}\mathbf{x} = \lambda\|\mathbf{x}\|^2.$$

In generale, per un vettore arbitrario, la disuguaglianza di Rayleigh fornisce limiti ottimali per il rapporto $\frac{\mathbf{x}^\top\mathbf{Q}\mathbf{x}}{\|\mathbf{x}\|^2}$.

Lemma 5.8.9. *Sia $\mathbf{Q} \in \mathbb{R}^{n\times n}$ tale che $\mathbf{Q} = \mathbf{Q}^\top \succ 0$. Allora la funzione*

$$V(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\top\mathbf{Q}(\mathbf{x} - \mathbf{x}^*)$$

è tale che $\sqrt{V(\mathbf{x})}$ è proporzionale alla distanza da $\mathbf{x}^*$. In particolare, esistono costanti $c_1, c_2 > 0$ tali che:

$$c_1\|\mathbf{x} - \mathbf{x}^*\|^2 \leq V(\mathbf{x}) \leq c_2\|\mathbf{x} - \mathbf{x}^*\|^2$$

per ogni $\mathbf{x} \in \mathbb{R}^n$.

Di conseguenza

$$V(\mathbf{x}_k) \to 0 \qquad \textit{se e solo se} \qquad \mathbf{x}_k \to \mathbf{x}^*.$$

In parole povere, $V(\mathbf{x})$ è proporzionale alla distanza al quadrato dal punto minimo $\mathbf{x}^*$.

Dimostrazione. Per la disuguaglianza di Rayleigh applicata al vettore $(\mathbf{x} - \mathbf{x}^*)$, abbiamo

$$\lambda_{\min}(\mathbf{Q})\|\mathbf{x} - \mathbf{x}^*\|^2 \leq (\mathbf{x} - \mathbf{x}^*)^\top\mathbf{Q}(\mathbf{x} - \mathbf{x}^*) \leq \lambda_{\max}(\mathbf{Q})\|\mathbf{x} - \mathbf{x}^*\|^2.$$

Moltiplicando tutti i membri per $\frac{1}{2}$ si ottiene

$$\frac{\lambda_{\min}(\mathbf{Q})}{2}\|\mathbf{x} - \mathbf{x}^*\|^2 \leq V(\mathbf{x}) \leq \frac{\lambda_{\max}(\mathbf{Q})}{2}\|\mathbf{x} - \mathbf{x}^*\|^2.$$

Ponendo

$$c_1 = \frac{\lambda_{\min}(\mathbf{Q})}{2} \qquad \text{e} \qquad c_2 = \frac{\lambda_{\max}(\mathbf{Q})}{2},$$

otteniamo la disuguaglianza desiderata.
In particolare $V(\mathbf{x}) \to 0$ se e solo se $\|\mathbf{x}_k - \mathbf{x}^*\| \to 0$, cioè

$$\mathbf{x}_k \to \mathbf{x}^*.$$

□

Enunciamo ora alcune stime che saranno utili per la dimostrazione del Teorema 5.8.11.

Lemma 5.8.10. *Per ogni $x \in [0,1)$ vale*

$$x \leq -\ln(1-x) \leq \frac{x}{1-x}.$$

In particolare, se $x \in [0, \frac{1}{2}]$ allora

$$x \leq -\ln(1-x) \leq 2x.$$

Dimostrazione. Dimostriamo prima la disuguaglianza inferiore

$$x \leq -\ln(1-x).$$

Consideriamo la funzione $f(t) = -\ln(1-t)$, definita per $t \in [0,1)$. Le sue prime due derivate sono

$$f'(t) = \frac{1}{1-t},$$
$$f''(t) = \frac{1}{(1-t)^2} > 0.$$

Dato che la derivata seconda è positiva la funzione f è convessa.
Per una funzione convessa, il grafico si trova sopra le sue tangenti. La tangente a f in $t = 0$ è:

$$T(t) = f(0) + f'(0)(t-0) = t.$$

Abbiamo quindi che per convessità

$$f(t) \geq T(t) \qquad \text{per ogni } t \in [0,1),$$

cioè

$$-\ln(1-t) \geq t$$

che è esattamente la disuguaglianza cercata.

Ora ci rimane di dimostrare la disuguaglianza superiore

$$-\ln(1-x) \leq \frac{x}{1-x}$$

Per fare ciò usiamo una funzione ausiliaria. Definiamo

$$g(x) = \frac{x}{1-x} + \ln(1-x).$$

Vogliamo mostrare che

$$g(x) \geq 0 \qquad \text{per ogni } x \in [0,1).$$

Calcoliamo la derivata

$$g'(x) = \frac{(1-x) - x(-1)}{(1-x)^2} - \frac{1}{1-x} = \frac{1}{(1-x)^2} - \frac{1}{1-x}$$
$$= \frac{1-(1-x)}{(1-x)^2} = \frac{x}{(1-x)^2}.$$

Per $x \geq 0$, abbiamo quindi che $g'(x) \geq 0$, ciò significa che g è crescente per $x \geq 0$.
Perciò

$$\frac{x}{1-x} + \ln(1-x) \geq 0 \quad \Longrightarrow \quad \frac{x}{1-x} \geq -\ln(1-x).$$

Nel caso in cui $x \in [0, \frac{1}{2}]$, sappiamo già che $-\ln(1-x) \leq \frac{x}{1-x}$. Se $x \in [0, \frac{1}{2}]$, allora $1 - x \geq \frac{1}{2}$, da cui

$$\frac{1}{1-x} \leq 2 \quad \Longrightarrow \quad \frac{x}{1-x} \leq 2x.$$

Quindi

$$-\ln(1-x) \leq 2x.$$

□

Il seguente teorema fornisce una condizione necessaria e sufficiente affinché, nel caso delle funzioni quadratiche, la sequenza $\{\mathbf{x}_k\}$ generata da un metodo di discesa del gradiente converga al punto di minimo $\mathbf{x}^*$, cioè

$$\lim_{k \to +\infty} \mathbf{x}_k = \mathbf{x}^*.$$

Teorema 5.8.11 (**Condizione di convergenza globale**). *Sia* $\{\mathbf{x}_k\}$ *la sequenza generata da un algoritmo di gradiente*

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k,$$

dove $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ *e sia* γ_k *definita come nel Lemma 5.8.4, supponendo che* $\gamma_k > 0$ *per ogni* k. *Allora la sequenza* $\{\mathbf{x}_k\}$ *converge a* $\mathbf{x}^*$ *per ogni condizione iniziale* $\mathbf{x}_0$ *se e solo se*

$$\sum_{k=0}^{+\infty} \gamma_k = +\infty.$$

Dimostrazione. Dal Lemma 5.8.4, abbiamo una formula che descrive come evolve V ad ogni iterazione

$$V(\mathbf{x}_{k+1}) = (1 - \gamma_k) V(\mathbf{x}_k)$$

dove $\gamma_k \in (0, 1]$, in virtù delle ipotesi e dell'Osservazione 5.8.5. D'altra parte, $\gamma_j = 1$ per qualche $j \in \mathbb{N}$ se e solo se $\mathbf{g}_j = \nabla f(\mathbf{x}_j) = 0$, ovvero se la successione ha raggiunto il punto stazionario $\mathbf{x}^*$, e dunque $\mathbf{x}_k = \mathbf{x}^*$ per ogni $k \geq j$. In tal caso, ovviamente $\mathbf{x}_k \to \mathbf{x}^*$, dato che, da un certo punto in poi, $\{\mathbf{x}_k\}$ è una successione costante, e chiaramente la serie dei γ_k diverge, dato che $\gamma_k = 1$ per ogni $k \geq j$. Quindi, possiamo supporre senza perdita di generalità che $\gamma_k \in (0, 1)$ per ogni $k \in \mathbb{N}$.
Applicando la formula ricorsiva ripetutamente a partire da $k = 0$, otteniamo un'espressione per $V(\mathbf{x}_k)$ in forma chiusa

$$V(\mathbf{x}_k) = V(\mathbf{x}_0) \prod_{i=0}^{k-1} (1 - \gamma_i).$$

Questo prodotto mostra che la funzione di errore decresce ad ogni passo, ma il passo di diminuzione dipende dal prodotto dei fattori $(1 - \gamma_i)$. Grazie al Lemma 5.8.9, abbiamo che

$$V(\mathbf{x}_k) \to 0 \quad \text{se e solo se} \quad \mathbf{x}_k \to \mathbf{x}^*.$$

Pertanto, per dimostrare che $\mathbf{x}_k \to \mathbf{x}^*$, dobbiamo dimostrare che

$$\lim_{k \to +\infty} V(\mathbf{x}_k) = 0.$$

Grazie all'equazione ricorsiva, questo avviene se e solo se

$$\lim_{k \to +\infty} \prod_{i=0}^{k-1} (1 - \gamma_i) = 0.$$

Applicando il logaritmo naturale, possiamo trasformare il prodotto in una serie

$$\ln\left(\prod_{i=0}^{k-1} (1 - \gamma_i) \right) = \sum_{i=0}^{k-1} \ln(1 - \gamma_i).$$

Prendendo il limite per $k \to +\infty$, il prodotto tende a zero se e solo se la sua somma logaritmica tende a $-\infty$:

$$\ln\Big(\prod_{i=0}^{k-1}(1-\gamma_i)\Big) \to -\infty \quad \iff \quad \sum_{i=0}^{+\infty}\ln(1-\gamma_i) = -\infty.$$

Moltiplicando per -1, questa condizione equivale a

$$\sum_{i=0}^{+\infty} -\ln(1-\gamma_i) = +\infty.$$

Abbiamo cosi riformulato il problema. La convergenza avviene se e solo se la serie $\sum_{i=0}^{+\infty} -\ln(1-\gamma_i)$ diverge.

Dobbiamo quindi dimostrare che

$$\sum_{i=0}^{+\infty} -\ln(1-\gamma_i) = +\infty \quad \iff \quad \sum_{i=0}^{+\infty}\gamma_i = +\infty.$$

Dimostriamo prima

$$\sum_{i=0}^{+\infty}\gamma_i = +\infty \quad \implies \quad \sum_{i=0}^{+\infty} -\ln(1-\gamma_i) = +\infty.$$

Per fare ciò, utilizziamo il Lemma 5.8.10:

$$\ln(t) \le t-1, \quad \forall t > 0.$$

Poniamo $t = 1-\gamma_i$. Poiché $\gamma_i \in (0,1)$, abbiamo $t \in (0,1)$ e la disuguaglianza vale. Sostituendo otteniamo:

$$\ln(1-\gamma_i) \le -\gamma_i,$$

il che implica

$$-\ln(1-\gamma_i) \ge \gamma_i.$$

Di conseguenza, se la serie $\sum_{i=0}^{+\infty}\gamma_i$ diverge, allora anche $\sum_{i=0}^{+\infty} -\ln(1-\gamma_i)$ deve divergere, poiché ogni suo termine è maggiore o uguale al corrispondente γ_i.

Viceversa, ora dimostriamo che

$$\sum_{i=0}^{+\infty}\gamma_i < +\infty \quad \implies \quad \sum_{i=0}^{+\infty} -\ln(1-\gamma_i) < +\infty.$$

Poiché $\sum_{i=0}^{+\infty}\gamma_i$ converge, il termine generale deve tendere a zero. Dunque esiste $N \in \mathbb{N}$ tale che per ogni $i \ge N$ vale $\gamma_i \le \frac{1}{2}$.

Per il Lemma 5.8.10, quando $\gamma_i \in [0, \frac{1}{2}]$, abbiamo

$$-\ln(1-\gamma_i) \le 2\gamma_i.$$

Quindi per $i \ge N$

$$\sum_{i=N}^{+\infty} -\ln(1-\gamma_i) < 2\sum_{i=N}^{+\infty}\gamma_i < +\infty.$$

Aggiungendo i primi N termini (che sono un numero finito), la serie completa $\sum_{i=0}^{+\infty} -\ln(1-\gamma_i)$ converge. □

Osservazione 5.8.12. L'ipotesi che $\gamma_k > 0$ è significativa, poiché garantisce la proprietà di discendenza dell'algoritmo: ad ogni passo il valore della funzione obiettivo diminuisce. Se tale condizione non è verificata, il teorema non è più valido e la convergenza potrebbe fallire.

5.8.3 Convergenza per i vari algoritmi della discesa del gradiente

Nella seguente analisi ci servirà sia la disuguaglianza che il Quoziente di Rayleigh.

Lemma 5.8.13 (**Quoziente di Rayleigh**). *Sia* $\mathbf{Q} = \mathbf{Q}^\top \succ 0$ *una matrice* $n \times n$*, allora per ogni* $\mathbf{x} \in \mathbb{R}^n$*,* $\mathbf{x} \neq 0$ *vale*

$$\frac{\lambda_{\min}(\mathbf{Q})}{\lambda_{\max}(\mathbf{Q})} \leq \frac{(\mathbf{x}^\top \mathbf{x})^2}{(\mathbf{x}^\top \mathbf{Q}\mathbf{x})(\mathbf{x}^\top \mathbf{Q}^{-1}\mathbf{x})} \leq \frac{\lambda_{\max}(\mathbf{Q})}{\lambda_{\min}(\mathbf{Q})}, \tag{5.8.7}$$

dove

$$R(\mathbf{x}) = \frac{(\mathbf{x}^\top \mathbf{x})^2}{(\mathbf{x}^\top \mathbf{Q}\mathbf{x})(\mathbf{x}^\top \mathbf{Q}^{-1}\mathbf{x})},$$

viene chiamato Quoziente di Rayleigh.

Dimostrazione. La chiave di questa dimostrazione è usare le forme della disuguaglianza di Rayleigh introdotte nella Proposizione 5.8.7

$$\lambda_{\min}(\mathbf{Q})\|\mathbf{x}\|^2 \leq \mathbf{x}^\top \mathbf{Q}\mathbf{x} \leq \lambda_{\max}(\mathbf{Q})\|\mathbf{x}\|^2,$$
$$\lambda_{\min}(\mathbf{Q}^{-1})\|\mathbf{x}\|^2 \leq \mathbf{x}^\top \mathbf{Q}^{-1}\mathbf{x} \leq \lambda_{\max}(\mathbf{Q}^{-1})\|\mathbf{x}\|^2.$$

e la relazione tra gli autovalori di $\mathbf{Q}$ e $\mathbf{Q}^{-1}$

$$\lambda_{\min}(\mathbf{Q}^{-1}) = \frac{1}{\lambda_{\max}(\mathbf{Q})},$$
$$\lambda_{\max}(\mathbf{Q}^{-1}) = \frac{1}{\lambda_{\min}(\mathbf{Q})}.$$

Dimostriamo prima la stima dall'alto

$$R(\mathbf{x}) = \frac{(\mathbf{x}^\top \mathbf{x})^2}{(\mathbf{x}^\top \mathbf{Q}\mathbf{x})(\mathbf{x}^\top \mathbf{Q}^{-1}\mathbf{x})} \leq \frac{\lambda_{\max}(\mathbf{Q})}{\lambda_{\min}(\mathbf{Q})}.$$

Otteniamo ciò maggiorando il denominatore e minorando il numeratore.
Minoriamo i denominatori:

$$\mathbf{x}^\top \mathbf{Q}\mathbf{x} \geq \lambda_{\min}(\mathbf{Q})\|\mathbf{x}\|^2,$$
$$\mathbf{x}^\top \mathbf{Q}^{-1}\mathbf{x} \geq \lambda_{\min}(\mathbf{Q}^{-1})\|\mathbf{x}\|^2 = \frac{1}{\lambda_{\max}(\mathbf{Q})}\|\mathbf{x}\|^2.$$

Facciamo il prodotto:

$$(\mathbf{x}^\top \mathbf{Q}\mathbf{x})(\mathbf{x}^\top \mathbf{Q}^{-1}\mathbf{x}) \geq \lambda_{\min}(\mathbf{Q})\|\mathbf{x}\|^2 \frac{1}{\lambda_{\max}(\mathbf{Q})}\|\mathbf{x}\|^2 = \frac{\lambda_{\min}(\mathbf{Q})}{\lambda_{\max}(\mathbf{Q})}\|\mathbf{x}\|^4.$$

Ora prendiamo il rapporto:

$$R(\mathbf{x}) = \frac{\|\mathbf{x}\|^4}{(\mathbf{x}^\top \mathbf{Q}\mathbf{x})(\mathbf{x}^\top \mathbf{Q}^{-1}\mathbf{x})} \leq \frac{\|\mathbf{x}\|^4}{\frac{\lambda_{\min}(\mathbf{Q})}{\lambda_{\max}(\mathbf{Q})}\|\mathbf{x}\|^4} = \frac{\lambda_{\min}(\mathbf{Q})}{\lambda_{\max}(\mathbf{Q})}.$$

Ora dimostriamo il limite inferiore. Stavolta facciamo l'opposto di prima, ovvero minoriamo il denominatore e maggioriamo il numeratore:

$$\mathbf{x}^\top \mathbf{Q}\mathbf{x} \leq \lambda_{\max}(\mathbf{Q})\|\mathbf{x}\|^2,$$
$$\mathbf{x}^\top \mathbf{Q}^{-1}\mathbf{x} \leq \lambda_{\max}(\mathbf{Q}^{-1})\|\mathbf{x}\|^2 = \frac{1}{\lambda_{\min}(\mathbf{Q})}\|\mathbf{x}\|^2.$$

Facciamo il prodotto:

$$(\mathbf{x}^\top \mathbf{Q}\mathbf{x})(\mathbf{x}^\top \mathbf{Q}^{-1}\mathbf{x}) \leq \lambda_{\max}(\mathbf{Q})\|\mathbf{x}\|^2 \cdot \frac{1}{\lambda_{\min}(\mathbf{Q})}\|\mathbf{x}\|^2 = \frac{\lambda_{\max}(\mathbf{Q})}{\lambda_{\min}(\mathbf{Q})}\|\mathbf{x}\|^4.$$

Ora prendiamo il rapporto:

$$R(\mathbf{x}) = \frac{\|\mathbf{x}\|^4}{(\mathbf{x}^\top \mathbf{Q}\mathbf{x})(\mathbf{x}^\top \mathbf{Q}^{-1}\mathbf{x})} \geq \frac{\|\mathbf{x}\|^4}{\frac{\lambda_{\max}(\mathbf{Q})}{\lambda_{\min}(\mathbf{Q})}\|\mathbf{x}\|^4} = \frac{\lambda_{\max}(\mathbf{Q})}{\lambda_{\min}(\mathbf{Q})}.$$

□

Ora abbiamo tutti gli strumenti per analizzare la convergenza della del metodo della discesa più ripida nel caso delle funzioni quadratiche.

Convergenza della discesa più ripida

Lemma 5.8.14. *Per il metodo del gradiente applicato alla funzione quadratica*

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{b}^\top \mathbf{x} + c,$$

con $\mathbf{Q} \in \mathbb{R}^{n\times n}$ simmetrica e definita positiva, il passo ottimale ad ogni iterazione (discesa più ripida) è dato da:

$$\alpha_k^{SD} = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k},$$

dove $\mathbf{g}_k = \nabla f(\mathbf{x}_k) = \mathbf{Q}\mathbf{x}_k - \mathbf{b}$.

Inoltre, con questa scelta del passo, il coefficiente di riduzione γ_k del Lemma 5.8.4 diventa:

$$\gamma_k = \frac{(\mathbf{g}_k^\top \mathbf{g}_k)^2}{(\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k)(\mathbf{g}_k^\top \mathbf{Q}^{-1}\mathbf{g}_k)}.$$

In particolare, abbiamo $0 < \gamma_k \leq 1$.

Dimostrazione. Per determinare il passo ottimale, consideriamo la funzione obiettivo lungo la direzione del gradiente:

$$\phi_k(\alpha) = f(\mathbf{x}_k - \alpha \mathbf{g}_k).$$

Sviluppiamo $\phi_k(\alpha)$:

$$\begin{aligned}\phi_k(\alpha) &= \frac{1}{2}(\mathbf{x}_k - \alpha\mathbf{g}_k)^\top \mathbf{Q}(\mathbf{x}_k - \alpha\mathbf{g}_k) - \mathbf{b}^\top(\mathbf{x}_k - \alpha\mathbf{g}_k) + c \\ &= \frac{1}{2}\mathbf{x}_k^\top \mathbf{Q}\mathbf{x}_k - \alpha\mathbf{g}_k^\top \mathbf{Q}\mathbf{x}_k + \frac{1}{2}\alpha^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k - \mathbf{b}^\top \mathbf{x}_k + \alpha\mathbf{b}^\top \mathbf{g}_k + c.\end{aligned}$$

Osserviamo che $\mathbf{g}_k = \mathbf{Q}\mathbf{x}_k - \mathbf{b}$, quindi:

$$\begin{aligned}\mathbf{g}_k^\top \mathbf{Q}\mathbf{x}_k &= (\mathbf{Q}\mathbf{x}_k - \mathbf{b})^\top \mathbf{Q}\mathbf{x}_k = \mathbf{x}_k^\top \mathbf{Q}^2\mathbf{x}_k - \mathbf{b}^\top \mathbf{Q}\mathbf{x}_k, \\ \mathbf{b}^\top \mathbf{g}_k &= \mathbf{b}^\top(\mathbf{Q}\mathbf{x}_k - \mathbf{b}) = \mathbf{b}^\top \mathbf{Q}\mathbf{x}_k - \|\mathbf{b}\|^2.\end{aligned}$$

Sostituendo otteniamo:

$$\begin{aligned}\phi_k(\alpha) = \Big[\frac{1}{2}\mathbf{x}_k^\top \mathbf{Q}\mathbf{x}_k - \mathbf{b}^\top \mathbf{x}_k + c\Big] - \alpha(\mathbf{x}_k^\top \mathbf{Q}^2\mathbf{x}_k - \mathbf{b}^\top \mathbf{Q}\mathbf{x}_k) \\ + \frac{1}{2}\alpha^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k + \alpha(\mathbf{b}^\top \mathbf{Q}\mathbf{x}_k - \|\mathbf{b}\|^2).\end{aligned}$$

Raggruppando i termini, notiamo che la parte tra parentesi quadre è proprio $f(\mathbf{x}_k)$, quindi:

$$\phi_k(\alpha) = f(\mathbf{x}_k) - \alpha \mathbf{g}_k^\top \mathbf{g}_k + \frac{1}{2}\alpha^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k,$$

dove abbiamo usato il fatto che

$$\mathbf{g}_k^\top \mathbf{g}_k = (\mathbf{Q}\mathbf{x}_k - \mathbf{b})^\top (\mathbf{Q}\mathbf{x}_k - \mathbf{b}) = \mathbf{x}_k^\top \mathbf{Q}^2 \mathbf{x}_k - 2\mathbf{b}^\top \mathbf{Q}\mathbf{x}_k + \|\mathbf{b}\|^2.$$

Per trovare il minimo di $\phi_k(\alpha)$, calcoliamo la derivata e la poniamo uguale a zero:

$$\phi_k'(\alpha) = -\mathbf{g}_k^\top \mathbf{g}_k + \alpha \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k = 0.$$

Risolvendo per α, otteniamo:

$$\alpha_k^{\mathrm{SD}} = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}.$$

Verifichiamo che si tratti effettivamente di un minimo calcolando la derivata seconda:

$$\phi''(\alpha) = \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k > 0,$$

dove la disuguaglianza stretta segue dal fatto che $\mathbf{Q}$ è definita positiva e $\mathbf{g}_k \neq \mathbf{0}$ (altrimenti saremmo già al minimo).

Per dimostrare la seconda parte del lemma, utilizziamo il risultato del Lemma 5.8.4. Sostituendo $\alpha_k = \alpha_k^{\mathrm{SD}}$ nell'espressione di γ_k:

$$\begin{aligned}
\gamma_k &= \frac{\alpha_k \mathbf{g}_k^\top \mathbf{g}_k - \frac{1}{2}\alpha_k^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}{V(\mathbf{x}_k)} \\
&= \frac{\frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k} \mathbf{g}_k^\top \mathbf{g}_k - \frac{1}{2}\left(\frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}\right)^2 \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}{V(\mathbf{x}_k)} \\
&= \frac{\frac{(\mathbf{g}_k^\top \mathbf{g}_k)^2}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k} - \frac{1}{2}\frac{(\mathbf{g}_k^\top \mathbf{g}_k)^2}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}}{V(\mathbf{x}_k)} = \frac{\frac{1}{2}\frac{(\mathbf{g}_k^\top \mathbf{g}_k)^2}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}}{V(\mathbf{x}_k)}.
\end{aligned}$$

Ricordiamo che $V(\mathbf{x}_k) = \frac{1}{2}(\mathbf{x}_k - \mathbf{x}^*)^\top \mathbf{Q}(\mathbf{x}_k - \mathbf{x}^*)$. Poiché $\mathbf{g}_k = \mathbf{Q}(\mathbf{x}_k - \mathbf{x}^*)$, abbiamo $\mathbf{x}_k - \mathbf{x}^* = \mathbf{Q}^{-1}\mathbf{g}_k$, quindi:

$$V(\mathbf{x}_k) = \frac{1}{2}(\mathbf{Q}^{-1}\mathbf{g}_k)^\top \mathbf{Q}(\mathbf{Q}^{-1}\mathbf{g}_k) = \frac{1}{2}\mathbf{g}_k^\top \mathbf{Q}^{-1}\mathbf{Q}\mathbf{Q}^{-1}\mathbf{g}_k = \frac{1}{2}\mathbf{g}_k^\top \mathbf{Q}^{-1}\mathbf{g}_k.$$

Sostituendo nell'espressione di γ_k:

$$\gamma_k = \frac{\frac{1}{2}\frac{(\mathbf{g}_k^\top \mathbf{g}_k)^2}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}}{\frac{1}{2}\mathbf{g}_k^\top \mathbf{Q}^{-1}\mathbf{g}_k} = \frac{(\mathbf{g}_k^\top \mathbf{g}_k)^2}{(\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k)(\mathbf{g}_k^\top \mathbf{Q}^{-1}\mathbf{g}_k)}.$$

Infine, osserviamo che γ_k è il Quoziente di Rayleigh applicato a $\mathbf{g}_k$, e dunque $\gamma_k > 0$, mentre per l'Osservazione 5.8.5 sappiamo già che $\gamma_k \leq 1$. □

Teorema 5.8.15. *Nel metodo della discesa più ripida applicato a una funzione quadratica, abbiamo $\mathbf{x}_k \to \mathbf{x}^*$ per ogni punto iniziale $\mathbf{x}_0$.*

Dimostrazione. Cominciamo con il caso banale.
Se per qualche k abbiamo $\mathbf{g}_k = 0$, allora $\mathbf{x}_k = \mathbf{x}^*$ e l'algoritmo termina.

Assumiamo ora invece che $\mathbf{g}_k \neq 0$ per tutti i k.
Riprendiamo la forma di γ_k da (5.8.5), che per il metodo di discesa più ripida è

$$\gamma_k = \alpha_k \frac{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}^{-1}\mathbf{g}_k}\left(2\frac{\mathbf{g}_k^\top\ \mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k} - \alpha_k\right).$$

Per il Lemma 5.8.14 sappiamo che il valore ottimale di α_k è dato da

$$\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}.$$

Ancora grazie al Lemma 5.8.14 sappiamo che

$$\gamma_k = \frac{\left(\mathbf{g}_k^\top \mathbf{g}_k\right)^2}{(\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k)(\mathbf{g}_k^\top \mathbf{Q}^{-1}\mathbf{g}_k)}.$$

Dimostriamo ora che γ_k è limitato inferiormente da una costante positiva.
Dato che $\gamma_k = R(\mathbf{g}_k)$, dove R è il Quoziente di Rayleigh, allora per il Lemma 5.8.13 otteniamo

$$\mathbf{g}_k \geq \frac{\lambda_{\min}(\mathbf{Q})}{\lambda_{\max}(\mathbf{Q})} > 0.$$

Abbiamo così dimostrato che per ogni k:

$$\gamma_k \geq \frac{\lambda_{\min}(\mathbf{Q})}{\lambda_{\max}(\mathbf{Q})} > 0.$$

Dunque, concludiamo che

$$\sum_{k=0}^{+\infty} \gamma_k = +\infty,$$

dato che il termine generale della serie è positivo e non va a zero. Per il Teorema 5.8.11, ciò implica immediatamente che $\mathbf{x}_k \to \mathbf{x}^*$ per $k \to +\infty$. □

Osservazione 5.8.16. In alternativa, si può concludere la dimostrazione del Teorema 5.8.15 senza ricorrere al Teorema 5.8.11, ma dimostrando direttamente la convergenza.

Dal Lemma 5.8.4 abbiamo:

$$V(\mathbf{x}_{k+1}) = (1 - \gamma_k)V(\mathbf{x}_k).$$

Poiché $\gamma_k \geq c > 0$ per ogni k, abbiamo $0 \leq 1 - \gamma_k \leq 1 - c < 1$. Iterando la relazione:

$$\begin{aligned} V(\mathbf{x}_1) &= (1 - \gamma_0)V(\mathbf{x}_0) \leq (1 - c)V(\mathbf{x}_0), \\ V(\mathbf{x}_2) &= (1 - \gamma_1)V(\mathbf{x}_1) \leq (1 - c)^2 V(\mathbf{x}_0), \\ &\vdots \\ V(\mathbf{x}_k) &\leq (1 - c)^k V(\mathbf{x}_0). \end{aligned}$$

Dato che $0 < 1 - c < 1$, abbiamo $(1 - c)^k \to 0$ per $k \to \infty$, e quindi:

$$\lim_{k\to\infty} V(\mathbf{x}_k) = 0.$$

Ricordando che $V(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^\top \mathbf{Q}(\mathbf{x}-\mathbf{x}^*)$ e che $\mathbf{Q}$ è definita positiva, esiste una costante $\mu > 0$ tale che:

$$V(\mathbf{x}) \geq \frac{\mu}{2}\|\mathbf{x}-\mathbf{x}^*\|^2 \quad \text{per ogni } \mathbf{x}.$$

Quindi:

$$0 \leq \|\mathbf{x}_k - \mathbf{x}^*\|^2 \leq \frac{2}{\mu}V(\mathbf{x}_k) \to 0 \quad \text{per } k \to \infty,$$

da cui segue:

$$\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x}^*.$$

Learning rate fisso

Consideriamo ora il metodo del gradiente con learning rate α fissato. Quindi

$$\alpha_k = \alpha > 0 \qquad \forall\ k \in \mathbb{N}.$$

L'algoritmo risultante è

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k).$$

Il grande vantaggio di questo algoritmo è la sua semplicità, non serve nessun line search per ogni iterazione per trovare α_k.
Di conseguenza però, la convergenza del algoritmo dipende dalla scelta di α. Non funziona per qualsiasi α. Il seguente teorema ci fornisce la condizione necessaria e sufficiente su α per la convergenza dell'algoritmo nel caso quadratico.

Teorema 5.8.17 (**Convergenza del gradiente con learning rate fisso - caso quadratico**). *Sia*

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{b}^\top \mathbf{x} + c$$

una funzione quadratica con $\mathbf{Q} = \mathbf{Q}^\top \succ 0$ *e sia* $\mathbf{x}^*$ *il suo unico minimo.*

Consideriamo il metodo del gradiente con learning rate α *costante*

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k).$$

Allora $\mathbf{x}_k \to \mathbf{x}^*$ *per ogni scelta del punto iniziale* $\mathbf{x}_0$ *se*

$$0 < \alpha < \frac{2}{\lambda_{\max}(\mathbf{Q})}. \tag{5.8.8}$$

Dimostrazione. Per dimostrare la convergenza $\mathbf{x}_k \to \mathbf{x}^*$, vogliamo sfruttare il Teorema 5.8.11. Grazie al Lemma 5.8.4, sappiamo che

$$\gamma_k = \alpha \frac{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}^{-1}\mathbf{g}_k}\left(2\frac{\mathbf{g}_k^\top\ \mathbf{g}_k}{\mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k} - \alpha\right),$$

dove

$$\mathbf{g}_k = \nabla f(\mathbf{x}_k) = \mathbf{Q}(\mathbf{x}_k - \mathbf{x}^*).$$

Per applicare il Teorema 5.8.11 mostriamo che $\gamma_k \geq c > 0$ per ogni k (a meno del caso banale in cui $\mathbf{g}_k = 0$, in cui la convergenza è già verificata), così che la serie $\sum_{k=0}^{\infty} \gamma_k$ diverga.
Grazie alle stime della Proposizione 5.8.7 e del Lemma 5.8.13, sappiamo che

$$\lambda_{\min}(\mathbf{Q})\|\mathbf{g}_k\|^2 \leq \mathbf{g}_k^\top \mathbf{Q}\mathbf{g}_k \leq \lambda_{\max}(\mathbf{Q})\|\mathbf{g}_k\|^2$$

e

$$\mathbf{g}_k^\top \mathbf{Q}^{-1}\mathbf{g}_k \geq \lambda_{\min}(\mathbf{Q}^{-1})\|\mathbf{g}_k\|^2 = \frac{1}{\lambda_{\max}(\mathbf{Q})}\|\mathbf{g}_k\|^2.$$

Dunque, per ogni $\alpha > 0$ ne segue che

$$\gamma_k \geq \alpha\lambda_{\min}(\mathbf{Q})^2 \left(2\frac{1}{\lambda_{\max}(\mathbf{Q})} - \alpha\right).$$

Quindi, se

$$\frac{2}{\lambda_{\max}(\mathbf{Q})} - \alpha > 0 \iff \alpha < \frac{2}{\lambda_{\max}(\mathbf{Q})},$$

allora concludiamo che

$$\gamma_k \geq c = \alpha\lambda_{\min}(\mathbf{Q})^2 \left(2\frac{1}{\lambda_{\max}(\mathbf{Q})} - \alpha\right) > 0.$$

Perciò applichiamo il Teorema 5.8.11 per concludere che

$$\mathbf{x}_k \to \mathbf{x}^* \qquad \text{per} \qquad k \to +\infty.$$

□

5.9 I Tipi di Gradient Descent: Compromesso tra Accuratezza e Velocità

L'addestramento di un modello di machine learning avviene, nella maggior parte dei casi, attraverso l'ottimizzazione di una funzione di costo (o loss function) mediante l'algoritmo del gradient descent. La modalità con cui vengono aggiornati i parametri del modello riveste un ruolo fondamentale e dipende fortemente dalla quantità di dati disponibili. Esistono tre principali varianti di questo algoritmo, ciascuna caratterizzata da un diverso equilibrio tra accuratezza computazionale, stabilità e velocità di convergenza.

5.9.1 Batch Gradient Descent

Il Batch Gradient Descent, noto anche come Vanishing Gradient Descent, rappresenta la formulazione più classica e intuitiva dell'algoritmo. Ad ogni iterazione, l'aggiornamento dei parametri è calcolato utilizzando l'intero training set. In altre parole, si valuta il gradiente della funzione di costo per ogni esempio di addestramento e si calcola la media di tutti i contributi, che costituisce la direzione di discesa per l'aggiornamento dei pesi.

Vantaggi:

- **Stabilità e Precisione:** Poiché utilizza l'intero data-set, la direzione del gradiente è una stima molto accurata del gradiente reale della funzione di costo. Ciò garantisce una discesa stabile e monotona verso il minimo (globale, in caso di funzioni convesse).
- **Convergenza Teoricamente Garantita:** Con un learning rate adeguato, il metodo converge con certezza al minimo della funzione.

Svantaggi:

- **Inefficienza Computazionale:** Per data-set di grandi dimensioni, il calcolo del gradiente su tutti i dati a ogni passo risulta estremamente oneroso in termini di tempo e risorse.
- **Elevato Consumo di Memoria:** L'intero data-set deve essere caricato in memoria, condizione spesso impraticabile nei casi reali.
- **Sensibilità ai Minimi Locali:** Nelle funzioni non convesse, l'uso della media globale del gradiente può portare a stagnazioni in minimi locali poco profondi.

In sintesi, il Batch Gradient Descent fornisce una traiettoria di discesa regolare e precisa, ma la sua lentezza lo rende poco adatto a data-set di grandi dimensioni.

5.9.2 Stochastic Gradient Descent

Per superare i limiti di efficienza del Batch Gradient Descent, lo Stochastic Gradient Descent adotta un approccio opposto: ad ogni iterazione, il gradiente viene calcolato su un singolo esempio di training, selezionato in modo casuale, e i parametri vengono aggiornati immediatamente.

Vantaggi:

- **Velocità ed Efficienza:** L'aggiornamento basato su un singolo dato è estremamente rapido e rende l'algoritmo adatto a data-set di grandi dimensioni.
- **Capacità di Evitare Minimi Locali:** La componente stocastica del gradiente permette all'algoritmo di uscire da minimi locali e di esplorare aree più ampie dello spazio dei parametri.

Svantaggi:

- **Alta Varianza e Instabilità:** Il gradiente stimato su un solo esempio è molto rumoroso, e ciò provoca oscillazioni significative nella funzione di costo.
- **Convergenza Imperfetta:** L'algoritmo tende a oscillare attorno al minimo invece di convergere esattamente.

In generale, lo Stochastic Gradient Descent sacrifica stabilità e precisione per una maggiore velocità di convergenza e una migliore capacità di esplorazione.

5.9.3 Mini-Batch Gradient Descent: il Compromesso Ottimale

Il Mini-Batch Gradient Descent rappresenta un compromesso efficace tra le due strategie precedenti, ed è oggi l'approccio più utilizzato nel deep learning. In questo caso, ad ogni iterazione si calcola il gradiente su un piccolo sottoinsieme del data-set, detto mini-batch, generalmente composto da 32 a 256 esempi.

Vantaggi:

- **Riduzione della Varianza:** Rispetto allo Stochastic Gradient Descent, l'uso di un mini-batch fornisce una stima più stabile del gradiente, producendo una discesa più regolare verso il minimo.
- **Efficienza Computazionale:** L'algoritmo può sfruttare appieno le operazioni vettorizzate e le architetture parallele delle GPU, migliorando sensibilmente le prestazioni.
- **Gestione della Memoria:** La dimensione del batch può essere adattata alle risorse disponibili, mantenendo un buon equilibrio tra efficienza e stabilità.

Svantaggi:

- **Iperparametro Aggiuntivo:** L'introduzione della dimensione del batch aggiunge un ulteriore parametro da calibrare. Batch troppo piccoli producono rumore, mentre batch eccessivamente grandi rallentano la convergenza.
- **Complessità Implementativa:** È necessaria una logica di batching dei dati.

Il Mini-Batch Gradient Descent combina la stabilità del Batch Gradient Descent con la velocità dell'Stochastic Gradient Descent, risultando lo standard di fatto per l'addestramento su larga scala.

5.9.4 Confronto e Scelta della Strategia

La scelta della variante più appropriata dipende dal contesto applicativo e dalle risorse disponibili. Il Batch Gradient Descent offre la traiettoria più regolare ma è limitato dalla sua lentezza. Lo Stochastic Gradient Descent è rapido e permette una migliore esplorazione, ma a costo di maggiore instabilità. Il Mini-Batch Gradient Descent emerge come la soluzione più equilibrata, combinando efficienza, stabilità e scalabilità, ed è per questo la scelta più comune negli algoritmi di apprendimento profondo. Comprendere tali compromessi è fondamentale per progettare strategie di ottimizzazione efficaci e modelli di apprendimento robusti.

Capitolo 6

Backpropagation

Dopo aver analizzato la struttura e il funzionamento delle reti neurali artificiali, affrontiamo ora il meccanismo matematico che consente loro di apprendere dai dati: l'algoritmo di backpropagation. Esso costituisce il cuore del processo di apprendimento supervisionato, poiché permette di calcolare in modo efficiente il gradiente della funzione di costo rispetto a tutti i parametri della rete. La backpropagation, combinata con la discesa del gradiente, fornisce quindi il collegamento tra la teoria dell'ottimizzazione e il comportamento adattivo delle reti neurali.

Notazione 6.0.1. In questo documento adottiamo la seguente notazione per descrivere i layer di una rete neurale feed-forward:

- $l \in \{1, 2, \dots, L\}$ indica un layer generico della rete,
- L indica l'output layer (l'ultimo strato),
- il layer di input è considerato il layer 0, e denotiamo il vettore di input con $\mathbf{x}$.

6.1 Apprendimento della rete

6.1.1 Addestramento e apprendimento

Abbiamo già menzionato che una rete neurale "impara" a svolgere un compito. Dal punto di vista matematico, l'apprendimento corrisponde alla determinazione automatica dei parametri $\theta = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^{(L)}$ in modo che la rete approssimi una funzione sconosciuta che governa i dati.

Ciò che distingue una rete neurale da un algoritmo tradizionale è che non le forniamo un insieme esplicito di istruzioni che stabiliscono come deve comportarsi. Al contrario, la rete apprende automaticamente dai dati.

Per ottenere questo risultato, è necessario disporre di un insieme di addestramento (o training set), tipicamente costituito da coppie

$$(\mathbf{x}_i, \hat{\mathbf{y}}_i) \in \mathbb{R}^{n_0} \times \mathbb{R}^{n_L}, \qquad i = 1, \dots, m,$$

dove $\mathbf{x}_i$ rappresenta l'input e $\hat{\mathbf{y}}_i$ il corrispondente valore desiderato (o target).

Definizione 6.1.1 (**Dati Etichettati**)**.** Nell'apprendimento supervisionato, ogni dato di input ha un'etichetta (o label) che rappresenta la risposta corretta.

Ad esempio, per addestrare una rete a riconoscere numeri scritti a mano, il training set conterrà immagini di cifre con l'etichetta corrispondente al numero rappresentato.

Il comportamento della rete è determinato dai valori dei pesi $\mathbf{W}^{(l)}$ e dei bias $\mathbf{b}^{(l)}$. I pesi regolano l'intensità delle connessioni tra neuroni di layer consecutivi, mentre i bias indicano la propensione dei neuroni ad attivarsi.

Inizialmente, i parametri vengono inizializzati con valori casuali, pertanto le prestazioni della rete sono scarse. Per guidare l'apprendimento, viene definita una funzione di costo (o di perdita, loss function) che misura la discrepanza tra le previsioni della rete e i valori target:

$$C(\theta) = \frac{1}{m} \sum_{i=1}^{m} \ell\Big(F(\mathbf{x}_i; \theta),\, \hat{\mathbf{y}}_i\Big),$$

dove $\ell(\cdot, \cdot)$ è una funzione di costo elementare.

Durante l'addestramento, la rete elabora i dati del training set e aggiorna iterativamente i parametri al fine di minimizzare $C(\theta)$. In questo modo, la rete apprende i pattern presenti nei dati osservati. Una volta addestrata, la funzione $F(\cdot; \theta)$ così ottenuta è in grado di generalizzare, ossia di produrre previsioni accurate su input non presenti nel training set.

6.2 Connessione con il Gradient Descent

Per poter addestrare una rete neurale è necessario disporre non solo di una funzione di costo $C(\theta)$, ma anche di un metodo efficace per ridurla. Il metodo della discesa del gradiente (gradient descent) fornisce un quadro generale per l'ottimizzazione, ma non specifica come calcolare in modo efficiente le derivate della funzione di costo rispetto a tutti i parametri della rete. Proprio per questo entra in gioco la backpropagation, che rappresenta il meccanismo pratico per ottenere tali derivate in reti di grandi dimensioni.

Il metodo del gradient descent richiede tre elementi fondamentali per poter essere applicato:

1. una funzione obiettivo C, che deve essere differenziabile rispetto ai parametri del modello, poiché il metodo si basa sul calcolo delle derivate;

2. il gradiente della funzione obiettivo rispetto a tutti i parametri (pesi e bias) della rete;

3. un learning rate $\alpha > 0$, che controlla l'ampiezza dei passi effettuati durante l'ottimizzazione.

Il gradiente della funzione obiettivo rispetto ai parametri indica in quale direzione occorre modificare tali parametri per ridurre il valore di C. Formalmente, possiamo scriverlo come:

$$\nabla C(\mathbf{W}^{(L)}, \mathbf{b}^{(L)}, \dots) = \Big(\dots, \frac{\partial C}{\partial \omega_{ij}^{(l)}}, \frac{\partial C}{\partial b_j^{(l)}}, \dots \Big),$$

dove ogni componente rappresenta la derivata parziale della funzione obiettivo rispetto a un singolo peso o bias della rete neurale.

Tali derivate esprimono quanto varia la funzione obiettivo al variare di ciascun parametro: in altre parole, quanto la costo aumenta o diminuisce se modifichiamo leggermente un determinato peso o bias. Conoscendo queste variazioni, possiamo quindi aggiornare i parametri nella direzione che riduce il valore di C.

6.2.1 Calcolo del gradiente

È proprio in questo punto che entra in gioco la backpropagation. Per reti neurali di grandi dimensioni, il numero di parametri è elevatissimo, e risulta impraticabile calcolare esplicitamente

tutte le derivate parziali. La backpropagation fornisce un procedimento sistematico ed efficiente per ottenere il gradiente ∇C applicando ripetutamente la regola della catena del calcolo differenziale, propagando all'indietro l'errore dalla funzione obiettivo verso i layer precedenti della rete.

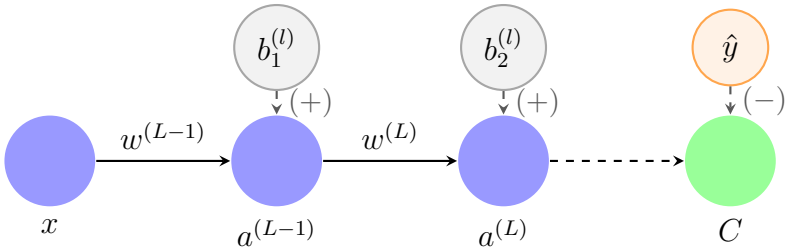
6.3 L'algoritmo di Backpropagation

L'obiettivo del machine learning consiste nell'ottimizzazione di una funzione di costo, la quale misura quanto l'uscita della rete si discosta dal valore desiderato. Ad ogni iterazione, la rete "impara", aggiusta i pesi e i bias per minimizzare tale funzione.

Per minimizzare la funzione di costo dobbiamo determinare quali dei pesi e dei bias dobbiamo aggiustare. Per fare ciò si utilizza l'algoritmo di discesa del gradiente, che aggiorna i pesi nella direzione opposta al gradiente della funzione di costo, ovvero nella direzione in cui essa diminuisce più rapidamente.

Per comprendere intuitivamente il funzionamento della backpropagation, analizziamo il caso più semplice possibile: una rete neurale con un unico neurone per strato, in questo caso non abbiamo a fare con formule vettoriali, ma solo con quelle scalari.

Nella figura vediamo una rappresentazione grafica. Abbiamo l'input layer x, un hidden layer $a^{(L-1)}$ e l'output layer $a^{(L)}$. Il valore del output layer viene poi usato per calcolare la funzione di costo.



Supponiamo di avere una funzione di costo semplice

$$C = (a^{(L)} - \hat{y})^2,$$

dove

- $a^{(L)}$ è l'attivazione in uscita,
- $\hat{y}$ è il valore desiderato (label).

Nel forward pass, se denotiamo con l un layer qualsiasi, la rete calcola

$$z^{(l)} = w^{(l)} a^{(l-1)} + b^{(l)},$$
$$a^{(l)} = f(z^{(l)}),$$

dove f è una funzione di attivazione generica, e l'apice l indica il layer nel quale ci troviamo. Abbiamo quindi, come sempre

$$\textbf{peso} \times \textbf{input} + \textbf{bias}$$

che poi passa attraverso la funzione di attivazione.

6.3.1 Caso base

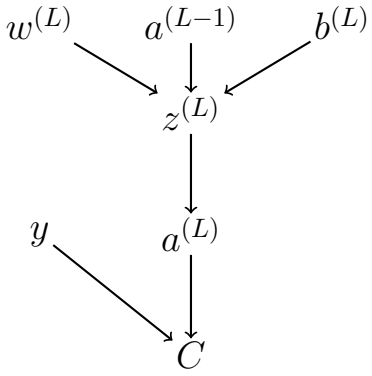
Per comprendere come varia la funzione di costo al variare di un singolo peso $\omega^{(L)}$, calcoliamo la derivata parziale:

$$\frac{\partial C}{\partial \omega^{(L)}}.$$

Ma il costo non dipende direttamente dal peso $\omega^{(L)}$, dipende in prima linea da $a^{(L)}$ che dipende da $z^{(L)}$ che poi a sua volta dipende da $\omega^{(L)}$ e $b^{(L)}$.
Applicando la regola della catena, per il peso che collega hidden layer a output layer otteniamo

$$\frac{\partial C}{\partial \omega^{(L)}} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial \omega^{(L)}}.$$

Specialmente per reti con più layer e neuroni può essere utile rappresentare questa struttura in un diagramma ad albero come in figura per tenere conto di cosa dipende da cosa.



Usando la definizione delle funzioni, possiamo calcolare le derivate parziali:

$$\begin{aligned} C = (a^{(L)} - y)^2 &\implies \frac{\partial C}{\partial a^{(L)}} = 2(a^{(L)} - y), \\ a^{(L)} = f(z^{(L)}) &\implies \frac{\partial a^{(L)}}{\partial z^{(L)}} = f'(z^{(L)}), \\ z^{(L)} = \omega^{(L)} a^{(L-1)} + b^{(L)} &\implies \frac{\partial z^{(L)}}{\partial \omega^{(L)}} = a^{(L-1)}. \end{aligned}$$

La nostra derivata dunque è

$$\frac{\partial C}{\partial \omega^{(L)}} = 2(a^{(L)} - y) f'(z^{(L)}) a^{(L-1)}.$$

Questa espressione indica quanto cambia l'errore complessivo se il peso $\omega^{(L)}$ viene modificato di una piccola quantità.

Anche i bias partecipano alla backpropagation in modo analogo ai pesi: a ciascun neurone corrisponde un termine di bias che viene aggiornato secondo lo stesso principio, ma con derivata parziale

$$z^{(L)} = \omega^{(L)} a^{(L-1)} + b^{(L)} \implies \frac{\partial z^{(L)}}{\partial b^{(L)}} = 1.$$

Otteniamo quindi come derivata:

$$\frac{\partial C}{\partial b^{(L)}} = 2(a^{(L)} - y) f'(z^{(L)}).$$

Questo è il cuore dell'algoritmo di backpropagation, che generalizza questo procedimento a tutti i pesi e bias della rete.

Se volessimo calcolare l'effetto di una variazione del peso $\omega^{(L-1)}$, dovremmo applicare un'altra catena di derivate, poiché l'errore non dipende direttamente da quei pesi, ma indirettamente tramite i layer successivi.
Quindi ora calcoliamo

$$\frac{\partial C}{\partial \omega^{(L-1)}}.$$

Supponiamo di avere una rete a tre livelli

$$\text{Input } (a^{(L-2)}) \quad \to \quad \text{Hidden } (a^{(L-1)}) \quad \to \quad \text{Output } (a^{(L)})$$

con le rispettive equazioni

$$\begin{aligned} z^{(L-1)} &= \omega^{(L-1)} a^{(L-2)} + b^{(L-1)}, \\ a^{(L-1)} &= f(z^{(L-1)}), \\ z^{(L)} &= \omega^{(L)} a^{(L-1)} + b^{(L)}, \\ a^{(L)} &= f(z^{(L)}), \\ C &= (a^{(L)} - y)^2. \end{aligned}$$

Vogliamo calcolare

$$\frac{\partial C}{\partial \omega^{(L-1)}}.$$

Poiché il costo non dipende direttamente da $w^{(L-1)}$, ma tramite i layer successivi, si applica la regola della catena:

$$\omega^{(L-1)} \to z^{(L-1)} \to a^{(L-1)} \to z^{(L)} \to a^{(L)} \to C.$$

Quindi dobbiamo applicare la regola della catena lungo questo percorso.

$$\frac{\partial C}{\partial \omega^{(L-1)}} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial \omega^{(L-1)}}.$$

Ora calcoliamo le varie derivate

$$\begin{aligned} C = (a^{(L)} - y)^2 &\implies \frac{\partial C}{\partial a^{(L)}} = 2(a^{(L)} - y), \\ a^{(L)} = f(z^{(L)}) &\implies \frac{\partial a^{(L)}}{\partial z^{(L)}} = f'(z^{(L)}), \\ z^{(L)} = \omega^{(L)} a^{(L-1)} + b^{(L)} &\implies \frac{\partial z^{(L)}}{\partial a^{(L-1)}} = \omega^{(L)}, \\ a^{(L-1)} = f(z^{(L-1)}) &\implies \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} = f'(z^{(L-1)}), \\ z^{(L-1)} = \omega^{(L-1)} a^{(L-2)} + b^{(L-1)} &\implies \frac{\partial z^{(L-1)}}{\partial \omega^{(L-1)}} = a^{(L-2)}. \end{aligned}$$

Moltiplichiamo tutto e otteniamo

$$\frac{\partial C}{\partial \omega^{(L-1)}} = 2(a^{(L)} - y) f'(z^{(L)}) \omega^{(L)} f'(z^{(L-1)}) a^{(L-2)}.$$

Questa formula mostra come la variazione del costo si propaga all'indietro fino al primo strato: modificando un peso dell'input layer, cambia l'attivazione dell'hidden layer, poi quella dell'output, e infine l'errore complessivo.

In generale, la backpropagation non fa altro che concatenare queste relazioni di causa-effetto tramite la regola della catena, calcolando le derivate parziali per tutti i pesi e bias della rete. Se si desidera calcolare la variazione del costo rispetto a un bias, si procede con lo stesso schema, sostituendo la derivata $\frac{\partial z}{\partial b} = 1$.

6.4 Generalizzazione a più livelli

Dagli esempi precedenti possiamo intuire che l'errore calcolato all'uscita della rete, $(\mathbf{a}^{(L)} - \hat{\mathbf{y}})^2$ (lavoriamo con vettori se abbiamo più livelli), viene propagato all'indietro attraverso i vari layer. Ad ogni layer, tale errore viene modulato dai pesi e dalla derivata della funzione di attivazione. In questo modo, l'informazione sull'errore "viaggia" a ritroso da un layer all'altro, venendo attenuata o amplificata in base ai parametri del modello.

In generale, abbiamo più neuroni per layer e ogni neurone riceve input da molti pesi e fornisce output a molti altri. La cosa importante, però, è che il principio della catena di derivate resta lo stesso: ogni variazione nel costo C si può esprimere come il prodotto di tutte le variazioni intermedie lungo il cammino che collega un peso all'uscita finale.
L'unica differenza è che e che ora ci sono tanti cammini paralleli, quindi l'errore si somma sui percorsi che passano per un dato neurone.

Nel caso generale, ciascun layer è composto da più neuroni: ogni neurone riceve input da molti pesi (collegati ai neuroni del layer precedente) e fornisce output a molti neuroni dello strato successivo.
Il principio matematico alla base, tuttavia, rimane invariato: secondo la regola della catena, ogni variazione della funzione di costo C rispetto a un peso può essere espressa come il prodotto delle variazioni intermedie lungo il percorso che collega tale peso all'uscita finale. Poiché ora esistono più percorsi paralleli, l'errore totale di un neurone si ottiene come somma degli errori propagati dai neuroni del layer successivo.

6.4.1 Notazione vettoriale

Consideriamo un layer generico l composto da n_l neuroni, tutti connessi ai neuroni del layer precedente $(l-1)$. Per un neurone qualsiasi j del layer l, il valore in ingresso (pre-attivazione) è dato da:

$$z_j^{(l)} = \sum_k \omega_{jk}^{(l)} a_k^{(l-1)} + b_j^{(l)}.$$

In altre parole, $z_j^{(l)}$ rappresenta la somma dei contributi in uscita dai neuroni del layer precedente $a_k^{(l-1)}$, ciascuno ponderato dal rispettivo peso $\omega_{jk}^{(l)}$, a cui si aggiunge il bias del neurone j-esimo $b_j^{(l)}$.
Il valore di uscita (post-attivazione) del neurone si ottiene applicando la funzione di attivazione $f(\cdot)$:

$$a_j^{(l)} = f(z_j^{(l)}).$$

In forma vettoriale, per l'intero layer si può scrivere:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)},$$
$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)}),$$

dove $\mathbf{W}^{(l)}$ è la matrice dei pesi, $\mathbf{a}^{(l-1)}$ è il vettore delle attivazioni in ingresso e $\mathbf{b}^{(l)}$ il vettore dei bias.

Esempio 6.4.1. Nel caso illustrato in Figura 6.4.1, in forma esplicita avremmo:

$$\begin{pmatrix} z_1^{(l)} \\ z_2^{(l)} \\ z_3^{(l)} \end{pmatrix} = \begin{bmatrix} \omega_{11}^{(l)} & \omega_{12}^{(l)} & \omega_{13}^{(l)} \\ \omega_{21}^{(l)} & \omega_{22}^{(l)} & \omega_{23}^{(l)} \\ \omega_{31}^{(l)} & \omega_{32}^{(l)} & \omega_{33}^{(l)} \end{bmatrix} \begin{pmatrix} a_1^{(l-1)} \\ a_2^{(l-1)} \\ a_3^{(l-1)} \end{pmatrix} + \begin{pmatrix} b_1^{(l)} \\ b_2^{(l)} \\ b_3^{(l)} \end{pmatrix}.$$

6.5 Errore locale e backpropagation

Il processo di apprendimento di una rete neurale si basa sulla minimizzazione di una funzione di costo C rispetto ai parametri della rete (pesi $\omega_{ij}^{(l)}$ e bias $b_j^{(l)}$). Il metodo più efficiente per calcolare il gradiente di C è l'algoritmo di backpropagation (Backpropagation), il cui concetto fondamentale è l'errore locale.

6.5.1 Definizione di errore locale

Possiamo ora introdurre formalmente il concetto di errore locale, che costituisce la quantità fondamentale per comprendere la propagazione dell'errore all'interno della rete. Attraverso di esso è possibile attribuire ad ogni neurone una "responsabilità" parziale per l'errore complessivo e determinare come questo errore debba essere retropropagato nei layer precedenti.

Nella rete neurale vogliamo calcolare le derivate

$$\frac{\partial C}{\partial \omega_{ij}^{(l)}} \quad \text{e} \quad \frac{\partial C}{\partial b_j^{(l)}},$$

cioè quanto la funzione obiettivo C cambi se modifichiamo di poco il peso o il bias di un certo neurone.

Durante l'addestramento, dopo che un input è stato processato dalla rete (fase di forward pass), si calcola l'errore all'output. L'idea centrale della backpropagation è di attribuire, o "propagare all'indietro", la responsabilità di questo errore a ciascun neurone nei layer interni.

Per quantificare questa responsabilità, si introduce la seguente definizione.

Definizione 6.5.1 (**Errore Locale**)**.** L'errore locale $\delta_j^{(l)}$ associato al neurone j dello strato l è definito come la derivata parziale della funzione di costo C rispetto all'input netto (o potenziale di attivazione) $z_j^{(l)}$ di quel neurone:

$$\delta_j^{(l)} := \frac{\partial C}{\partial z_j^{(l)}}.$$

Questa quantità misura la sensibilità del costo C rispetto a una piccola variazione del valore $z_j^{(l)}$.

La backpropagation ci darà la possibilità di calcolare $\delta_j^{(l)}$ per ogni layer e poi associarli agli errori nelle quantità di interesse, cioè $\frac{\partial C}{\partial \omega_{jk}^{(l)}}$ e $\frac{\partial C}{\partial b_{jk}^{(l)}}$.

L'interpretazione dell'errore locale è la seguente:

- se $\delta_j^{(l)} > 0$, un aumento di $z_j^{(l)}$ comporta un aumento del costo C;
- se $\delta_j^{(l)} < 0$, un aumento di $z_j^{(l)}$ comporta una diminuzione di C;
- il valore assoluto $|\delta_j^{(l)}|$ indica l'ampiezza di questa influenza. Un valore grande significa che piccole variazioni in $z_j^{(l)}$ provocano grandi cambiamenti in C.

Calcolare direttamente $\frac{\partial C}{\partial \omega_{ij}^{(l)}}$ per un peso in uno strato interno è complesso, poiché C dipende da $\omega_{ij}^{(l)}$ solo attraverso l'input netto $z_j^{(l)}$ e da tutti i successivi layer della rete. Tuttavia, utilizzando la regola della catena, si può esprimere la derivata del peso in termini dell'errore locale:

$$\frac{\partial C}{\partial \omega_{ij}^{(l)}} = \frac{\partial C}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial \omega_{ij}^{(l)}} = \delta_j^{(l)} a_i^{(l-1)}.$$

Pertanto, una volta noti gli errori locali $\delta_j^{(l)}$, il calcolo dei gradienti per i pesi e i bias diventa immediato. Il problema si riduce dunque a calcolare efficientemente $\delta_j^{(l)}$ per ogni neurone in ogni strato.

6.5.2 Derivazione della backpropagation nel caso scalare

Consideriamo inizialmente la rete semplificata con un solo neurone per strato di prima. Questo approccio ci permette di seguire la catena di dipendenze senza l'overhead notazionale degli indici.

La backpropagation si basa su quattro equazioni fondamentali. Insieme, queste equazioni ci forniscono un modo per calcolare sia l'errore che il gradiente della funzione di costo.

Errore nell'Output Layer

La rete produce un'output $a^{(L)} \in \mathbb{R}$, mentre il target è $\hat{y} \in \mathbb{R}$, e vogliamo minimizzare una funzione di costo $C(a^{(L)}, \hat{y})$ che misura la differenza tra output e target.

Durante la backpropagation vogliamo analizzare come la funzione costo varia rispetto ai parametri dell'ultimo strato (cioè $\omega^{(L)}$ e $b^{(L)}$). Per definizione l'errore locale dell'output layer è:

$$\delta^{(L)} = \frac{\partial C}{\partial z^{(L)}}.$$

Usando la regola della catena

$$\delta^{(L)} = \frac{\partial C}{\partial z^{(L)}} = \frac{\partial C}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}}.$$

Poiché $a^{(L)} = f(z^{(L)})$, abbiamo

$$\delta^{(L)} = \frac{\partial C}{\partial a^{(L)}} f'(z^{(L)}).$$

Il primo termine $\frac{\partial C}{\partial a^{(L)}}$ ci dice quanto velocemente il costo cambia in funzione dell'uscita del neurone nello strato L (ricordiamo che siamo nel caso scalare con un solo neurone per strato).

Come abbiamo visto in precedenza, tutti i termini in questa equazione sono facilmente calcolabili.

Errore $\delta^{(l)}$ in funzione dell'errore nel layer successivo $\delta^{(l+1)}$

Vediamo ora la seconda delle quattro equazioni base. L'input netto del neurone dello strato l, $z^{(l)}$ influenza il costo C nel seguente modo.
Prima determina l'attivazione del neurone nello strato:

$$a^{(l)} = f(z^{(l)}).$$

Questo a sua volta influenza l'input netto dello strato successivo:

$$z^{(l+1)} = \omega^{(l+1)} a^{(l)} + b^{(l+1)}.$$

Partendo sempre dalla definizione dell'errore locale

$$\delta^{(l)} = \frac{\partial C}{\partial z^{(l)}},$$

è necessario considerare entrambi gli effetti. Applicando la regola della catena attraverso lo strato successivo $(l+1)$, otteniamo:

$$\delta^{(l)} = \frac{\partial C}{\partial z^{(l)}} = \frac{\partial C}{\partial z^{(l+1)}} \frac{\partial z^{(l+1)}}{\partial z^{(l)}}.$$

Analizziamo i due fattori.
Il primo fattore è, per definizione, l'errore locale dello strato successivo:

$$\frac{\partial C}{\partial z^{(l+1)}} = \delta^{(l+1)}.$$

Il secondo fattore misura come $z^{(l)}$ influenza $z^{(l+1)}$. Per calcolarlo, ci serve l'output del neurone $a^{(l)}$:

$$\frac{\partial z^{(l+1)}}{\partial z^{(l)}} = \frac{\partial z^{(l+1)}}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}}.$$

Dalla definizione di $z^{(l+1)}$

$$z^{(l+1)} = \omega^{(l+1)} a_i^{(l)} + b^{(l+1)},$$

derivando rispetto a $a_j^{(l)}$

$$\frac{\partial z^{(l+1)}}{\partial a^{(l)}} = \omega^{(l+1)},$$

è il peso.
Dalla definizione di attivazione

$$a^{(l)} = f(z^{(l)}),$$

otteniamo

$$\frac{\partial a^{(l)}}{\partial z^{(l)}} = f'(z^{(l)}).$$

Sostituendo entrambi i risultati nell'Equazione (6.5.2), si ricava la formula del errore locale per il caso scalare:

$$\delta^{(l)} = \delta^{(l+1)} \omega^{(l+1)} f'(z^{(l)}).$$

Questa equazione mostra che l'errore locale allo strato l può essere calcolato ricorsivamente a partire dall'errore locale allo strato $l+1$. In altre parole, l'errore di ciascun neurone dipende da quello dei neuroni successivi, ponderato dai pesi e modulato dalla derivata della funzione di attivazione.

Equazione per il tasso di variazione del costo rispetto a qualsiasi bias nella rete

La terza equazione fondamentale ci dice come il costo cambia se cambiamo leggermente il bias di un certo layer l.

Anche il bias influenza il costo solo indirettamente:

$$b^{(l)} \longrightarrow z^{(l)} \longrightarrow a^{(l)} \longrightarrow z^{(l+1)} \longrightarrow a^{(l+1)} \longrightarrow \cdots \longrightarrow a^{(L)} \longrightarrow C.$$

Quindi come in precedenza dobbiamo applicare la regola della catena lungo tutta la rete

$$\frac{\partial C}{\partial b^{(l)}} = \frac{\partial C}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial b^{(l)}}.$$

Ma, per definizione $z^{(l)} = \omega^l a^{(l-1)} + b^{(l)}$, quindi

$$\frac{\partial z^{(l)}}{\partial b^{(l)}} = 1.$$

Perciò

$$\frac{\partial C}{\partial b^{(l)}} = \frac{\partial C}{\partial z^{(l)}} = \delta^{(l)},$$

dove il termine $\delta^{(l)}$ è proprio l'errore locale del neurone al layer l. Ciò significa che l'errore locale misura esattamente quanto il costo cambia se modifichiamo il bias in quel neurone.

Equazione per il tasso di variazione del costo rispetto a qualsiasi peso nella rete

L'ultima delle quattro equazioni fondamentali della backpropagation calcola

$$\frac{\partial C}{\partial \omega^{(l)}},$$

cioè come varia il costo totale se modifichiamo leggermente il peso $\omega^{(l)}$ di un certo layer intermedio.

Come già visto nei casi precedenti, il peso influenza il costo tramite una catena di dipendenze

$$\omega^{(l)} \longrightarrow z^{(l)} \longrightarrow a^{(l)} \longrightarrow z^{(l+1)} \longrightarrow a^{(l+1)} \longrightarrow \cdots \longrightarrow a^{(L)} \longrightarrow C.$$

Applicando la regola della catena otteniamo quindi

$$\frac{\partial C}{\partial \omega^{(l)}} = \frac{\partial C}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial \omega^{(l)}}.$$

Poiché per costruzione $z^{(l)} = \omega^l a^{(l-1)} + b^{(l)}$, abbiamo

$$\frac{\partial z^{(l)}}{\partial \omega^{(l)}} = a^{(l-1)}.$$

Definiamo come sempre l'errore locale del layer l come $\delta^{(l)} = \frac{\partial C}{\partial z^{(l)}}$, e otteniamo

$$\frac{\partial C}{\partial \omega^{(l)}} = \delta^{(l)} a^{(l-1)}.$$

Riassumendo abbiamo visto che le quattro equazioni fondamentali della backpropagation, nel caso scalare sono:

1. Errore output layer:
$$\delta^{(L)} = \frac{\partial C}{\partial a^{(L)}} f'(z^{(L)});$$
2. Errore layer qualsiasi:
$$\delta^{(l)} = \delta^{(l+1)} \omega^{(l+1)} f'(z^{(l)});$$
3. Gradiente rispetto ai bias
$$\frac{\partial C}{\partial b^{(l)}} = \frac{\partial C}{\partial z^{(l)}} = \delta^{(l)};$$
4. Gradiente rispetto ai pesi
$$\frac{\partial C}{\partial \omega^{(l)}} = \delta^{(l)} a^{(l-1)}.$$

Nella realtà non si usano mai reti cosi semplici, quindi dobbiamo generalizzare queste formule, e ricavare la loro forma vettoriale.

6.6 Forma vettoriale della backpropagation

Estendiamo ora il risultato al caso generale di una rete con più neuroni per strato. In questo contesto, lavoriamo con

- $\mathbf{a}^{(l)} \in \mathbb{R}^{n_l}$ vettore di attivazione del layer l;
- $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{(l-1)}}$ matrice dei pesi che collegano il layer $l-1$ al layer l;
- $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$ vettore dei bias.

La funzione di attivazione f agisce elemento per elemento e la funzione di costo è una funzione $C : \mathbb{R}^{n_L} \to \mathbb{R}$ che dipende da i valori in uscita del output layer $\mathbf{a}^{(L)}$ e dei valori target $\hat{\mathbf{y}} \in \mathbb{R}^{n_L}$.

6.6.1 Forward pass

Nel forward pass, l'informazione fluisce in avanti. Ogni neurone j del layer l contribuisce, con la propria attivazione $a_j^{(l)}$, a tutti i neuroni del layer successivo $l+1$, pesato da $\omega_{kj}^{(l+1)}$. Formalmente, questa relazione si scrive, se guardiamo il neurone k del layer $l+1$
$$z_k^{(l+1)} = \sum_j \omega_{kj}^{(l+1)} a_j^{(l)} + b_k^{(l+1)}.$$
Qui j varia sui neuroni in input (del layer l) mentre k rappresenta il nostro neurone in output. In forma matriciale
$$\mathbf{z}^{(l+1)} = \mathbf{W}^{(l+1)} \mathbf{a}^{(l)} + \mathbf{b}^{(l+1)},$$
$$\mathbf{a}^{(l)} = f\Big(\mathbf{z}^{(l)}\Big),$$
dove $\mathbf{W}^{(l+1)}$ è la matrice dei pesi che, agendo sul vettore delle attivazioni $\mathbf{a}^{(l)}$, genera il vettore dei potenziali $\mathbf{z}^{(l+1)}$ del layer successivo.

Dato che k è fisso e j varia sui neuroni in input, potremmo scrivere l'equazione scalare anche come
$$z_1^{(l+1)} = \begin{pmatrix} \omega_{11}^{(l+1)} & \omega_{12}^{(l+1)} & \dots & \omega_{1n_l}^{(l+1)} \end{pmatrix} \begin{pmatrix} a_1^{(l)} \\ a_2^{(l)} \\ \vdots \\ a_{n_l}^{(l)} \end{pmatrix} + b_1^{(l+1)}.$$
Ogni riga di $\mathbf{W}^{(l+1)}$ combina quindi le attivazione del layer precedente per calcolare un neurone del layer successivo.

6.6.2 Backward pass

Nel backward pass, invece, l'informazione viaggia al contrario, e l'obiettivo è calcolare

$$\frac{\partial C}{\partial \mathbf{W}^{(l)}} \quad \text{e} \quad \frac{\partial C}{\partial \mathbf{b}^{(l)}},$$

a partire dal errore locale $\boldsymbol{\delta}^{(l)}$ dei layer intermedi.
Per fare ciò ci servono le quattro equazioni fondamentali in forma vettoriale.

Errore nell'output layer

Ora abbiamo multiple neuroni nell'output layer. Per ogni neurone j-esimo dell'output layer vale la relazione scalare ricavata in precedenza:

$$\delta_j^{(L)} = \frac{\partial C}{\partial a_j^{(L)}} f'(z_j^{(L)}).$$

Scrivendo questo per tutti i neuroni, otteniamo

$$\begin{pmatrix} \delta_1^{(L)} \\ \delta_2^{(L)} \\ \vdots \\ \delta_{n_L}^{(L)} \end{pmatrix} = \begin{pmatrix} \frac{\partial C}{\partial a_1^{(L)}} f'(z_1^{(L)}) \\ \frac{\partial C}{\partial a_2^{(L)}} f'(z_2^{(L)}) \\ \vdots \\ \frac{\partial C}{\partial a_{n_L}^{(L)}} f'(z_{n_L}^{(L)}) \end{pmatrix}.$$

Osserviamo che

$$\nabla_{\mathbf{a}^{(L)}} C = \begin{pmatrix} \frac{\partial C}{\partial a_1^{(L)}} \\ \frac{\partial C}{\partial a_2^{(L)}} \\ \vdots \\ \frac{\partial C}{\partial a_{n_L}^{(L)}} \end{pmatrix}$$

e

$$f'(\mathbf{z}^{(L)}) = \begin{pmatrix} f'(z_1^{(L)}) \\ f'(z_2^{(L)}) \\ \vdots \\ f'(z_{n_L}^{(L)}) \end{pmatrix}.$$

Quindi, usando il prodotto di Hadamard ($\odot$), possiamo scrivere in forma vettoriale compatta:

$$\boldsymbol{\delta}^{(L)} = \nabla_{\mathbf{a}^{(L)}} C \odot f'(\mathbf{z}^{(L)}).$$

Propagazione dell'Errore all'indietro

Partiamo sempre dal caso scalare

$$\delta^{(l)} = \delta^{(l+1)} \omega^{(l+1)} f'(z^{(l)}).$$

Ora osserviamo che

- lo strato l ha n_l neuroni;
- lo strato $l+1$ ha $n_{(l+1)}$ neuroni;

- la matrice dei pesi $\mathbf{W}^{(l+1)} \in \mathbb{R}^{n_{+1} \times n_l}$.

In questo caso, per un neurone j nello strato l, l'errore $\delta_j^{(l)}$ dipende da tutti i neuroni dello strato $l+1$.
Ricordiamo la catena di dipendenze

$$z_j^{(l)} \longrightarrow a_j^{(l)} \longrightarrow \begin{pmatrix} z_1^{(l+1)} \\ z_2^{(l+1)} \\ \vdots \\ z_{n_{(l+1)}}^{(l+1)} \end{pmatrix} \longrightarrow \cdots \longrightarrow C.$$

Applicando la regola della catena per più variabili, dobbiamo sommare lungo tutti questi percorsi:

$$\delta_j^{(l)} = \frac{\partial C}{\partial z_j^{(l)}} = \sum_{k=1}^{n_{(l+1)}} \frac{\partial C}{\partial z_k^{(l+1)}} \frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}} \frac{\partial a_j^{(l)}}{\partial z_j^{(l)}}.$$

Analizziamo i singoli termini:

- per definizione
$$\frac{\partial C}{\partial z_k^{(l+1)}} = \delta_k^{(l+1)};$$
- dalla definizione $z_k^{(l+1)} = \sum_j \omega_{kj}^{(l+1)} a_j^{(l+1)} + b_k^{(l+1)}$ otteniamo che
$$\frac{\partial z_k^{(l+1)}}{\partial a_j^{(l)}} = \omega_{kj}^{(l+1)};$$
- sempre dalla definizione di $a_j^{(l)} = f(z_j^{(l)})$,
$$\frac{\partial a_j^{(l)}}{\partial z_j^{(l)}} = f'(z_j^{(l)}).$$

Quindi

$$\delta_j^{(l)} = \Big(\sum_{k=1}^{n_{(l+1)}} \delta_k^{(l+1)} \omega_{kj}^{(l+1)} \Big) f'(z_j^{(l)}).$$

Scrivendolo per tutti i neuroni $j = \{1, \ldots, n_l\}$

$$\begin{pmatrix} \delta_1^{(l)} \\ \delta_2^{(l)} \\ \vdots \\ \delta_{n_l}^{(l)} \end{pmatrix} = \begin{pmatrix} \Big(\sum_{k=1}^{n_{(l+1)}} \delta_k^{(l+1)} \omega_{k1}^{(l+1)} \Big) f'(z_1^{(l)}) \\ \Big(\sum_{k=1}^{n_{(l+1)}} \delta_k^{(l+1)} \omega_{k2}^{(l+1)} \Big) f'(z_2^{(l)}) \\ \vdots \\ \Big(\sum_{k=1}^{n_{(l+1)}} \delta_k^{(l+1)} \omega_{kn_l}^{(l+1)} \Big) f'(z_{n_l}^{(l)}) \end{pmatrix}.$$

Osserviamo che $\sum_{k=1}^{n_{(l+1)}} \delta_k^{(l+1)} \omega_{kj}^{(l+1)}$ è esattamente l'elemento j esimo del prodotto $(\mathbf{W}^{(l+1)})^\top \boldsymbol{\delta}^{(l+1)}$.
Infine, per incorporare la derivata della funzione di attivazione, si effettua un prodotto elemento per elemento (prodotto di Hadamard) con il vettore $f'(\mathbf{z}^{(l)})$.

Si arriva così alla forma vettoriale compatta e fondamentale dell'equazione di backpropagation:

$$\boldsymbol{\delta}^{(l)} = \Big(\mathbf{W}^{(l+1)\top} \boldsymbol{\delta}^{(l+1)} \Big) \odot f'(\mathbf{z}^{(l)}).$$

In forma compatta, l'errore si propaga all'indietro moltiplicando per la trasposta della matrice dei pesi: ciò riflette il flusso inverso dell'informazione.

Osservazione 6.6.1. Proviamo a capire meglio perché appare la trasposta della matrice pesi $\mathbf{W}^{(l+1)\top}$ nel backward pass. Stiamo calcolando l'errore locale $\delta_j^{(l)}$ per un neurone j nello strato l.

Riprendiamo l'operazione del Forward pass per un intero layer:

$$\mathbf{z}^{(l+1)} = \mathbf{W}^{(l+1)}\mathbf{a}^{(l)} + \mathbf{b}^{(l+1)}.$$

In questo caso $\mathbf{W}^{(l+1)}$ è una matrice $n_{(l+1)} \times n_l$, che viene moltiplicata per il vettore colonna $\mathbf{a}^{(l)}$ $(n_{(l+1)} \times 1)$ per produrre un vettore di dimensione $n_{(l+1)} \times 1$.
Abbiamo tante righe quanti sono i neuroni nello strato $l+1$, ogni riga j-esima (per $j \in \{0, 1, \ldots, n_{l+1}\}$) contiene tutti i pesi che arrivano al neurone j-esimo del layer $l+1$. Ogni riga rappresenta un neurone dello strato successivo.

Arriviamo al dunque: nel backward pass, vogliamo il vettore $\boldsymbol{\delta}^{(l)}$ a partire da $\boldsymbol{\delta}^{(l+1)}$. Guardando nuovamente la componente k-esima di $\boldsymbol{\delta}^{(l)}$:

$$\delta_j^{(l)} = \left(\sum_{k=1}^{n_{(l+1)}} \omega_{kj}^{(l+1)} \delta_k^{(l+1)} \right) f'(z_j^{(l)}).$$

Notiamo la somiglianza della sommatoria

$$\sum_{k=1}^{n_{(l+1)}} \omega_{kj}^{(l+1)} \delta_k^{(l+1)},$$

con un prodotto scalare. Ma, stavolta l'indice k scorre sulle righe di $\mathbf{W}^{(l+1)}$, mentre j è fisso. Questo significa che per un j fisso, stiamo prendendo tutti i pesi $\omega_{1j}^{(l+1)}, \omega_{2j}^{(l+1)}, \ldots, \omega_{n_{(l+1)}j}^{(l+1)}$. E questi elementi formano proprio l'intera colonna k-esima della matrice $\mathbf{W}^{(l+1)}$.
In altre parole, invece di guardare tutti i pesi che sono collegati a un neurone j-esimo nello strato $l+1$, guardiamo tutti i pesi attraverso i quali un neurone k dello strato l è connesso allo strato successivo.

Se vogliamo calcolare questa sommatoria per tutti i neuroni k, nello strato l simultaneamente, stiamo facendo la seguente operazione: per ogni colonna k di $\mathbf{W}^{(l+1)}$, facciamo il prodotto scalare con il vettore $\boldsymbol{\delta}^{(l+1)}$. E in algebra lineare questa operazione è esattamente la moltiplicazione della matrice trasposta per quel vettore.

Gradiente rispetto ai Bias

Dobbiamo applicare la formula del caso scalare a ogni neurone j nello strato l

$$\frac{\partial C}{\partial b_j^{(l)}} = \frac{\partial C}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial b_j^{(l)}}.$$

Poiché $z_j^{(l)} = \sum_k \omega_{jk}^{(l)} a_k^{(l-1)} + b_j^{(l)}$, abbiamo che

$$\frac{\partial z_j^{(l)}}{\partial b_j^{(l)}} = 1,$$

e per definizione $\frac{\partial C}{\partial z_j^{(l)}} = \delta_j^{(l)}$, perciò otteniamo

$$\frac{\partial C}{\partial b_j^{(l)}} = \delta_j^{(l)}.$$

Quindi per tutti i bias

$$\frac{\partial C}{\partial \mathbf{b}^{(l)}} = \begin{pmatrix} \frac{\partial C}{\partial b_1^{(l)}} \\ \frac{\partial C}{\partial b_2^{(l)}} \\ \vdots \\ \frac{\partial C}{\partial b_{n_l}^{(l)}} \end{pmatrix} = \begin{pmatrix} \delta_1^{(l)} \\ \delta_2^{(l)} \\ \vdots \\ \delta_{n_l}^{(l)} \end{pmatrix} = \boldsymbol{\delta}^{(l)}.$$

Gradiente rispetto ai pesi

Consideriamo il peso $\omega_{jk}^{(l)}$ che connette il neurone k dello strato $l-1$ al neurone j dello strato l.
La regola della catena ci dice

$$\frac{\partial C}{\partial \omega_{jk}^{(l)}} = \frac{\partial C}{\partial z_{jk}^{(l)}} \frac{\partial z_{jk}^{(l)}}{\partial \omega_{jk}^{(l)}} = \delta_j^{(l)} a_k^{(l-1)}.$$

Poiché $z_j^{(l)} = \sum_m \omega_{jm}^{(l)} a_m^{(l-1)} + b_j^{(l)}$, abbiamo

$$\frac{\partial z_j^{(l)}}{\partial \omega_{jk}^{(l)}} = a_k^{(l-1)}.$$

Ora, per tutti i pesi nella matrice $\mathbf{W}^{(l)}$ abbiamo

$$\frac{\partial C}{\partial \mathbf{W}^{(l)}} = \begin{bmatrix} \frac{\partial C}{\partial \omega_{11}^{(l)}} & \frac{\partial C}{\partial \omega_{12}^{(l)}} & \cdots & \frac{\partial C}{\partial \omega_{1n_{(l-1)}}^{(l)}} \\ \frac{\partial C}{\partial \omega_{21}^{(l)}} & \frac{\partial C}{\partial \omega_{22}^{(l)}} & \cdots & \frac{\partial C}{\partial \omega_{2n_{(l-1)}}^{(l)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial C}{\partial \omega_{n_l 1}^{(l)}} & \frac{\partial C}{\partial \omega_{n_l 2}^{(l)}} & \cdots & \frac{\partial C}{\partial \omega_{n_l n_{(l-1)}}^{(l)}} \end{bmatrix} = \begin{bmatrix} \delta_1^{(l)} a_1^{(l-1)} & \delta_1^{(l)} a_2^{(l-1)} & \cdots & \delta_1^{(l)} a_{n_{(l-1)}}^{(l-1)} \\ \delta_2^{(l)} a_1^{(l-1)} & \delta_2^{(l)} a_2^{(l-1)} & \cdots & \delta_2^{(l)} a_{n_{(l-1)}}^{(l-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{n_l}^{(l)} a_1^{(l-1)} & \delta_{n_l}^{(l)} a_2^{(l-1)} & \cdots & \delta_{n_l}^{(l)} a_{n_{(l-1)}}^{(l-1)} \end{bmatrix}.$$

E questa matrice è esattamente il prodotto esterno

$$\frac{\partial C}{\partial \mathbf{W}^{(l)}} = \boldsymbol{\delta}^{(l)} (\mathbf{a}^{(l-1)})^\top.$$

Definizione 6.6.2. Un prodotto esterno è un'operazione tra due vettori che produce una matrice.
Dati due vettori $\mathbf{u} \in \mathbb{R}^m$ e $\mathbf{v} \in \mathbb{R}^n$, il loro prodotto esterno è la matrice $m \times n$ data da

$$\mathbf{u}\mathbf{v}^\top = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{pmatrix} \begin{pmatrix} v_1 & v_2 & \dots & v_n \end{pmatrix} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & \dots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \dots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_m v_1 & u_m v_2 & \dots & u_m v_n \end{bmatrix}.$$

Osservazione 6.6.3. La forma vettoriale della backpropagation rivela una dualità elegante tra forward e backward pass:

- **Forward Pass**: La matrice $\mathbf{W}^{(l+1)}$ propaga in avanti le attivazioni, trasformando lo spazio degli output dello strato l nello spazio degli input dello strato $l+1$.
- **Backward Pass**: La matrice trasposta $\mathbf{W}^{(l+1)\top}$ propaga all'indietro gli errori, proiettando lo spazio delle sensibilità dello strato $l+1$ nello spazio delle sensibilità dello strato l.

Questa simmetria non è solo notazionale ma riflette la profonda connessione tra il flusso di informazione e il flusso del gradiente nella rete.

In conclusione, le quattro equazioni fondamentali della backpropagation nella loro forma vettoriale riassumono in modo compatto l'intero meccanismo di apprendimento di una rete neurale feed-forward:

$$\begin{aligned}
\boldsymbol{\delta}^{(L)} &= \nabla_{\mathbf{a}^{(L)}} C \odot f'(\mathbf{z}^{(L)}), \\
\boldsymbol{\delta}^{(l)} &= \left(\mathbf{W}^{(l+1)}\right)^{\top} \boldsymbol{\delta}^{(l+1)} \odot f'(\mathbf{z}^{(l)}), \\
\frac{\partial C}{\partial \mathbf{b}^{(l)}} &= \boldsymbol{\delta}^{(l)}, \\
\frac{\partial C}{\partial \mathbf{W}^{(l)}} &= \boldsymbol{\delta}^{(l)} (\mathbf{a}^{(l-1)})^{\top}.
\end{aligned}$$

Queste relazioni costituiscono la base per l'implementazione numerica della backpropagation e per tutti i moderni algoritmi di ottimizzazione basati sul gradiente.

6.7 Connessione con il Gradient Descent

Con l'algoritmo di backpropagation abbiamo trovato un modo efficiente per calcolare le derivate della funzione di costo C rispetto a tutti i parametri della rete:

$$\frac{\partial C}{\partial \mathbf{W}^{(l)}} \quad \text{e} \quad \frac{\partial C}{\partial \mathbf{b}^{(l)}}, \qquad \forall\, l = 1, \ldots, L.$$

Tuttavia, per aggiornare effettivamente i pesi e i bias durante l'addestramento, è necessario collegare la backpropagation a un metodo di ottimizzazione, come la discesa del gradiente.

Il gradient descent è un algoritmo di ottimizzazione che minimizza una funzione $C(\theta)$, aggiornando i parametri θ nella direzione opposta al gradiente:

$$\theta_{k+1} = \theta_k - \alpha \, \nabla_\theta C(\theta_k), \tag{6.7.1}$$

dove $\alpha > 0$ è la learning rate.

Nel caso di una rete neurale, i parametri θ sono tutti i pesi e bias

$$\theta = \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}, \ldots, \mathbf{W}^{(L)}, \mathbf{b}^{(L)}\}.$$

Il gradiente $\nabla_\theta C$ può contenere migliaia o milioni di derivate parziali. Calcolarlo manualmente diventa quindi quasi impossibile.

Applicando (6.7.1) a ciascun layer otteniamo:

$$\begin{aligned}
\mathbf{W}^{(l)} &\leftarrow \mathbf{W}^{(l)} - \alpha \, \frac{\partial C}{\partial \mathbf{W}^{(l)}}, \\
\mathbf{b}^{(l)} &\leftarrow \mathbf{b}^{(l)} - \alpha \, \frac{\partial C}{\partial \mathbf{b}^{(l)}}.
\end{aligned}$$

La backpropagation è il metodo per calcolare in modo efficiente queste derivate.
Il primo passo operativo della backpropagation è calcolare l'errore nel layer di uscita:

$$\delta^{(L)} = \frac{\partial C}{\partial \mathbf{a}^{(L)}} \odot f'(\mathbf{z}^{(L)}).$$

Partiamo quindi solo dall'ultimo layer, dove l'errore è direttamente osservabile confrontando l'output con la verità desiderata $\hat{\mathbf{y}}$.

Poi si propaga l'errore all'indietro verso i layer precedenti, usando la regola della catena. Per ogni layer intermedio calcoliamo

$$\delta^{(l)} = \left((\mathbf{W}^{(l+1)})^\top \boldsymbol{\delta}^{(l+1)}\right) \odot f'(\mathbf{z}^{(l)}).$$

Una volta noti tutti i $\boldsymbol{\delta}^{(l)}$, si possono calcolare i gradienti. Per ogni layer abbiamo

$$\frac{\partial C}{\partial \mathbf{W}^{(l)}} = \boldsymbol{\delta}^{(l)}(\mathbf{a}^{(l-1)})^\top,$$
$$\frac{\partial C}{\partial \mathbf{b}^{(l)}} = \boldsymbol{\delta}^{(l)}.$$

Questi gradienti sono poi usati dal gradient descent per aggiornare i parametri. Otteniamo quindi la regola di aggiornamento esplicita:

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \alpha\, \boldsymbol{\delta}^{(l)}(\mathbf{a}^{(l-1)})^\top,$$
$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \alpha\, \boldsymbol{\delta}^{(l)}.$$

Osservazione 6.7.1. Il metodo della discesa del gradiente spinge i parametri nella direzione opposta al gradiente della funzione di costo, riducendo iterativamente il valore di C. In questo contesto, la backpropagation non è un algoritmo di ottimizzazione, ma un meccanismo di calcolo del gradiente che rende possibile l'applicazione del gradient descent in reti con milioni di parametri.

6.8 Esempio Numerico

Presentiamo un esempio numerico completo con valori scelti per rendere più evidenti i meccanismi della backpropagation. I valori sono stati selezionati per accentuare le differenze tra i percorsi e rendere più visibili gli effetti dell'aggiornamento.

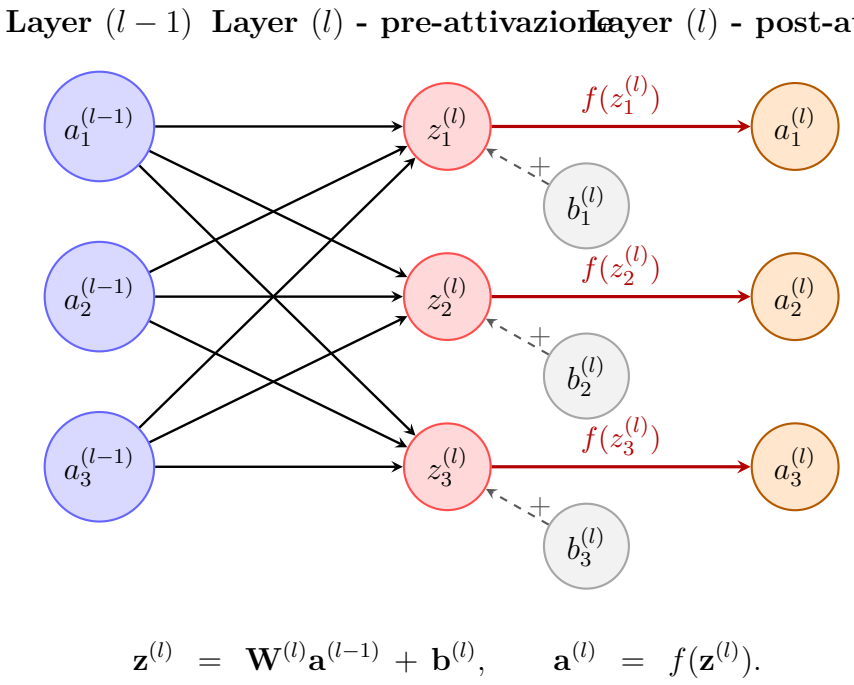
6.8.1 Architettura della Rete

Consideriamo una rete neurale con la seguente architettura (Figura 6.2):

6.8.2 Parametri e Dati

Definiamo i parametri della rete e i dati di input con valori che accentuano le differenze:

- **Pesi del layer nascosto**:
$$\mathbf{W}^{(1)} = \begin{bmatrix} 0.6 & 0.4 \\ 0.1 & 0.9 \end{bmatrix};$$
- **Pesi del layer di output**:
$$\mathbf{W}^{(2)} = \begin{bmatrix} 0.8 & 0.2 \end{bmatrix};$$
- **Bias**: Per semplicità, consideriamo tutti i bias nulli:
$$\mathbf{b}^{(1)} = \mathbf{0}, \quad b^{(2)} = 0;$$



$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \qquad \mathbf{a}^{(l)} = f(\mathbf{z}^{(l)}).$$

Figura 6.1: Rappresentazione del layer (l) diviso in due parti: nella fase di pre-attivazione $(z_j^{(l)})$ ogni neurone somma i contributi pesati dai neuroni del layer precedente e il proprio bias; nella fase di post-attivazione $(a_j^{(l)})$ il risultato viene trasformato tramite la funzione di attivazione f.

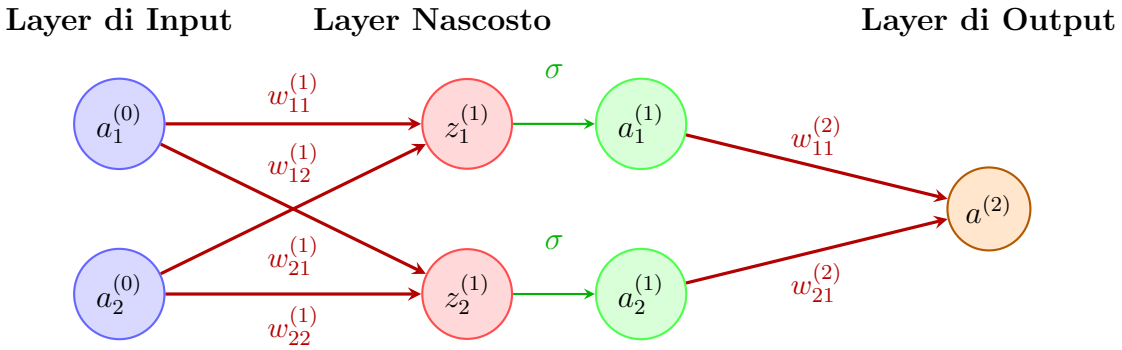


Figura 6.2: Architettura della rete neurale considerata nell'esempio

- **Input**:
$$\mathbf{x} = \begin{pmatrix} 0.8 \\ 0.2 \end{pmatrix};$$
- **Target**:
$$\hat{y} = 0.7;$$
- **Learning rate**:
$$\alpha = 0.5.$$

6.8.3 Funzioni di Attivazione e Costo

- **Funzione di attivazione**: Sigmoide
$$\sigma(z) = \frac{1}{1 + e^{-z}},$$
che ha derivata
$$\sigma'(z) = \sigma(z)(1 - \sigma(z)).$$
- **Funzione di costo**: Mean Squared Error (MSE)
$$C = \frac{1}{2}(a^{(L)} - \hat{y})^2.$$

6.8.4 Forward Pass

Layer Nascosto Calcoliamo prima i valori di pre-attivazione:

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} = \begin{bmatrix} 0.6 & 0.4 \\ 0.1 & 0.9 \end{bmatrix} \begin{pmatrix} 0.8 \\ 0.2 \end{pmatrix} = \begin{pmatrix} 0.6 \times 0.8 + 0.4 \times 0.2 \\ 0.1 \times 0.8 + 0.9 \times 0.2 \end{pmatrix} = \begin{pmatrix} 0.56 \\ 0.26 \end{pmatrix}.$$

Applicando la sigmoide, otteniamo:

$$a_1^{(1)} = \sigma(0.56) = \frac{1}{1 + e^{-0.56}} \approx 0.6368,$$
$$a_2^{(1)} = \sigma(0.26) = \frac{1}{1 + e^{-0.26}} \approx 0.5646.$$

Layer di Output

$$z^{(2)} = \mathbf{W}^{(2)}\mathbf{a}^{(1)} = \begin{pmatrix} 0.8 & 0.2 \end{pmatrix} \begin{pmatrix} 0.6368 \\ 0.5646 \end{pmatrix} = 0.8 \times 0.6368 + 0.2 \times 0.5646 = 0.6272,$$
$$a^{(2)} = \sigma(0.6272) = \frac{1}{1 + e^{-0.6272}} \approx 0.6518.$$

Calcolo dell'Errore

$$C = \frac{1}{2}(a^{(2)} - \hat{y})^2 = \frac{1}{2}(0.6518 - 0.7)^2$$
$$= \frac{1}{2}(-0.0482)^2 \approx 0.001161.$$

La Figura 6.3 mostra la rete con tutti i valori calcolati:

6.8.5 Backward Pass

Errore sul Neurone di Output Calcoliamo il gradiente rispetto all'input del neurone di output:

$$\begin{aligned}
\delta^{(2)} &= \frac{\partial C}{\partial z^{(2)}} = \frac{\partial C}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \\
&= (a^{(2)} - \hat{y}) \sigma'(z^{(2)}) \\
&= (a^{(2)} - \hat{y}) \Big(\sigma(z)(1 - \sigma(z)) \Big) \\
&= (0.6518 - 0.7) \times [0.6518 \times (1 - 0.6518)] \\
&= (-0.0482) \times 0.2269 \\
&\approx -0.01094.
\end{aligned}$$

Osservazione 6.8.1. Il termine $\delta^{(2)} = -0.01094$ è negativo, indicando che l'output della rete è inferiore al target. Per ridurre l'errore, dovremmo aumentare il valore di $z^{(2)}$, il che significa aumentare alcuni pesi del layer di output.

Propagazione dell'Errore al Layer Nascosto Per ogni neurone nel layer nascosto, calcoliamo l'errore propagato. Notiamo che i due neuroni ricevono contributi diversi a causa dei pesi differenti:

$$\begin{aligned}
\delta_1^{(1)} &= \frac{\partial C}{\partial z_1^{(1)}} = \sigma'(z_1^{(1)}) w_{11}^{(2)} \delta^{(2)} \\
&= [a_1^{(1)}(1 - a_1^{(1)})] w_{11}^{(2)} \delta^{(2)} \\
&= [0.6368 \times (1 - 0.6368)] \times 0.8 \times (-0.01094) \\
&= 0.2313 \times 0.8 \times (-0.01094) \\
&\approx -0.002024;
\end{aligned}$$

$$\begin{aligned}
\delta_2^{(1)} &= \sigma'(z_2^{(1)}) w_{21}^{(2)} \delta^{(2)} \\
&= [0.5646 \times (1 - 0.5646)] \times 0.2 \times (-0.01094) \\
&= 0.2457 \times 0.2 \times (-0.01094) \\
&\approx -0.000538.
\end{aligned}$$

Osservazione 6.8.2. Notiamo che $\delta_1^{(1)}$ è circa 4 volte più grande di $\delta_2^{(1)}$, riflettendo il fatto che il primo neurone nascosto ha un peso di output maggiore (0.8 vs 0.2) e quindi contribuisce di più all'errore finale.

Calcolo dei Gradienti

Gradienti per i pesi $\mathbf{W}^{(2)}$:

$$\begin{aligned}
\frac{\partial C}{\partial w_{11}^{(2)}} &= a_1^{(1)} \delta^{(2)} = 0.6368 \times (-0.01094) \approx -0.006966; \\
\frac{\partial C}{\partial w_{21}^{(2)}} &= a_2^{(1)} \delta^{(2)} = 0.5646 \times (-0.01094) \approx -0.006177.
\end{aligned}$$

Gradienti per i pesi $\mathbf{W}^{(1)}$:

$$\frac{\partial C}{\partial w_{11}^{(1)}} = x_1 \delta_1^{(1)} = 0.8 \times (-0.002024) \approx -0.001619;$$
$$\frac{\partial C}{\partial w_{12}^{(1)}} = x_1 \delta_2^{(1)} = 0.8 \times (-0.000538) \approx -0.000430;$$
$$\frac{\partial C}{\partial w_{21}^{(1)}} = x_2 \delta_1^{(1)} = 0.2 \times (-0.002024) \approx -0.000405;$$
$$\frac{\partial C}{\partial w_{22}^{(1)}} = x_2 \delta_2^{(1)} = 0.2 \times (-0.000538) \approx -0.000108.$$

6.8.6 Aggiornamento dei Pesi

Applichiamo la discesa del gradiente per aggiornare tutti i pesi:

Pesi del layer di output:

$$w_{11}^{(2)} \leftarrow w_{11}^{(2)} - \alpha \frac{\partial C}{\partial w_{11}^{(2)}} = 0.8 - 0.5 \times (-0.006966) \approx 0.80348;$$
$$w_{21}^{(2)} \leftarrow w_{21}^{(2)} - \alpha \frac{\partial C}{\partial w_{21}^{(2)}} = 0.2 - 0.5 \times (-0.006177) \approx 0.20309.$$

Pesi del layer nascosto:

$$w_{11}^{(1)} \leftarrow 0.6 - 0.5 \times (-0.001619) \approx 0.60081;$$
$$w_{12}^{(1)} \leftarrow 0.4 - 0.5 \times (-0.000430) \approx 0.40022;$$
$$w_{21}^{(1)} \leftarrow 0.1 - 0.5 \times (-0.000405) \approx 0.10020;$$
$$w_{22}^{(1)} \leftarrow 0.9 - 0.5 \times (-0.000108) \approx 0.90005.$$

6.8.7 Verifica del Miglioramento

Eseguiamo un nuovo forward pass con i pesi aggiornati per verificare la riduzione dell'errore:

Forward pass con pesi aggiornati:

$$z_1^{(1)} = 0.60081 \times 0.8 + 0.40022 \times 0.2 \approx 0.56069;$$
$$z_2^{(1)} = 0.10020 \times 0.8 + 0.90005 \times 0.2 \approx 0.26017;$$
$$a_1^{(1)} = \sigma(0.56069) \approx 0.6369;$$
$$a_2^{(1)} = \sigma(0.26017) \approx 0.5647;$$
$$z^{(2)} = 0.80348 \times 0.6369 + 0.20309 \times 0.5647 \approx 0.6321;$$
$$a^{(2)} = \sigma(0.6321) \approx 0.6532;$$
$$C_{\text{new}} = \frac{1}{2}(0.6532 - 0.7)^2 \approx 0.001096.$$

Osservazione 6.8.3. L'errore è diminuito da 0.001161 a 0.001096, con una riduzione relativa di circa 5.6%. L'output si è avvicinato al target da 0.6518 a 0.6532, confermando l'efficacia dell'aggiornamento.

6.8.8 Analisi dei Risultati

Distribuzione degli Aggiornamenti:

- **Layer di output**: I pesi sono aumentati (come atteso per un errore negativo), con l'aggiornamento più significativo per $w_{11}^{(2)}$ che ha il gradiente più grande in valore assoluto.
- **Layer nascosto**: Gli aggiornamenti sono più piccoli ma seguono un pattern interessante:
  - $w_{11}^{(1)}$ e $w_{12}^{(1)}$ (collegati a $x_1 = 0.8$) hanno aggiornamenti più grandi;
  - $w_{21}^{(1)}$ e $w_{22}^{(1)}$ (collegati a $x_2 = 0.2$) hanno aggiornamenti più piccoli;
  - questo riflette il fatto che x_1 contribuisce maggiormente all'output finale.

6.8.9 Conclusioni

Questo esempio con valori numericamente significativi dimostra chiaramente:

1. **Proporzionalità della backpropagation**: i gradienti sono proporzionali all'influenza di ciascun parametro sull'errore finale;
2. **Distribuzione intelligente dell'errore**: la backpropagation attribuisce correttamente più "colpa" ai pesi che contribuiscono maggiormente all'errore;
3. **Effetto delle attivazioni**: i neuroni con attivazioni più vicine a 0.5 (dove la sigmoide è più sensibile) contribuiscono maggiormente alla propagazione dell'errore;
4. **Scelta dei parametri**: valori ben scelti rendono più evidenti i meccanismi dell'apprendimento;
5. **Convergenza**: anche con un solo passo di aggiornamento, l'errore diminuisce significativamente.

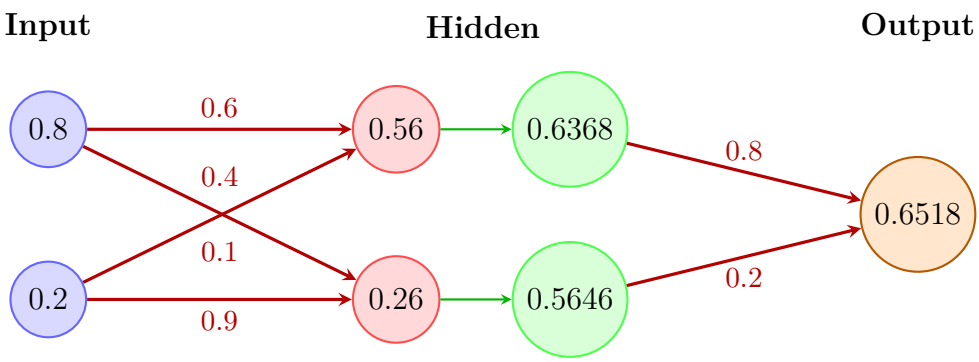


Figura 6.3: Rete neurale con valori numerici dopo il forward pass

Bibliografia

[1] Amir Ali Ahmadi. *ORF523 — Lecture 2: Mathematical Background*. Princeton University, 2016.

[2] Amir Ali Ahmadi. *ORF523 — Lecture 3: Local and Global Minima*. Princeton University, 2016.

[3] Amir Ali Ahmadi. *ORF523 — Lecture 4: Convex Sets and Functions*. Princeton University, 2016.

[4] A Comprehensive Guide on Deep Learning Optimizers. *Analytics Vidhya*. Available online: `https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/` (accessed 10 Nov 2025).

[5] Annette Klüx. *Backpropagation: Theory and Implementation*. Available online: `https://cklixx.people.wm.edu/teaching/math400/Annette-paper.pdf` (accessed 10 Nov 2025).

[6] *Advances in Optimization for Deep Learning*. arXiv preprint arXiv:2208.03897, 2022. Available online: `https://arxiv.org/pdf/2208.03897` (accessed 10 Nov 2025).

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] Edwin K. P. Chong and Stanislaw H. Zak. *An Introduction to Optimization*. Wiley, 4th edition, 2013.

[9] Giovanni Eugenio Comi. *Analisi Matematica B — Note delle lezioni*. Dispense del corso, Università di Bologna, 2024–2025.

[10] Stanford University. *CS231n — Lecture 4: Backpropagation*. Available online: `https://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture4.pdf` (accessed 10 Nov 2025).

[11] Mastering Backpropagation. *DataCamp Tutorials*. Available online: `https://www.datacamp.com/tutorial/mastering-backpropagation` (accessed 10 Nov 2025).

[12] Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning*. Cambridge University Press, 2020.

[13] Nicola Fusco, Paolo Marcellini e Carlo Sbordone. *Elementi di Analisi Matematica Due*. Liguori Editore, 2001.

[14] ETH Zürich. *Gradient Descent — Lecture 3*. Available online: `https://kyng.inf.ethz.ch/courses/AGA020/lectures/lecture3_gd.pdf` (accessed 10 Nov 2025).

[15] Georgia Tech. *Matrix Transformations*. Available online: `https://textbooks.math.gatech.edu/ila/matrix-transformations.html` (accessed 10 Nov 2025).

[16] What is Gradient Descent? *GeeksforGeeks — Data Science*. Available online: https://www.geeksforgeeks.org/data-science/what-is-gradient-descent/ (accessed 10 Nov 2025).

[17] Eigen-Decomposition of a Matrix. *GeeksforGeeks*. Available online: https://www.geeksforgeeks.org/engineering-mathematics/eigen-decomposition-of-a-matrix/ (accessed 10 Nov 2025).

[18] Optimization Rules in Deep Neural Networks. *GeeksforGeeks*. Available online: https://www.geeksforgeeks.org/deep-learning/optimization-rule-in-deep-neural-networks/ (accessed 10 Nov 2025).

[19] Quadratic Form of a Matrix. *GeeksforGeeks*. Available online: https://www.geeksforgeeks.org/engineering-mathematics/quadratic-form-of-a-matrix/ (accessed 10 Nov 2025).

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[21] Khan Academy. *Multivariable Chain Rule — Simple Version*. Available online: https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/differentiating-vector-valued-functions/a/multivariable-chain-rule-simple-version (accessed 10 Nov 2025).

[22] Johannes Lederer. *Activation Functions in Artificial Neural Networks: A Systematic Overview*. arXiv:2101.09957, 2021.

[23] Davide Maltoni. *Deep Learning — Introduzione*. Slide del corso di Pattern Recognition, Università di Bologna, 2024–2025.

[24] Davide Maltoni. *Deep Learning — Sequence to Sequence, RNN, LSTM, GRU*. Slide del corso di Pattern Recognition, Università di Bologna, 2024–2025.

[25] Math Insight. *Derivation: Directional Derivative and Gradient*. Available online: https://mathinsight.org/directional_derivative_gradient_derivation (accessed 10 Nov 2025).

[26] Abhay Singh. *Gradient Descent Explained — The Engine Behind AI Training*. Medium. Available online: https://medium.com/@abhaysingh71711/gradient-descent-explained-the-engine-behind-ai-training-2d8ef6ecad6f (accessed 10 Nov 2025).

[27] Backpropagation — Come apprende una rete neurale. *CannyDataScience / Medium*. Available online: https://cannydatascience.medium.com/backpropagation-come-apprende-una-rete-neurale-91c0c900fbc (accessed 10 Nov 2025).

[28] MIT OpenCourseWare. *Nonlinear Programming — Lecture 6: Constrained Optimization*. MIT, Spring 2004. Available online: https://ocw.mit.edu/courses/15-084j-nonlinear-programming-spring-2004/21073fc1bb9e5f68b3319bec0895683f_lec6_constr_opt.pdf (accessed 10 Nov 2025).

[29] Steven G. Johnson. *Norm Equivalence in Finite Dimensions*. MIT Applied Mathematics. Available online: https://math.mit.edu/~stevenj/18.335/norm-equivalence.pdf (accessed 10 Nov 2025).

[30] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[31] Michael A. Nielsen. *Chapter 2: How the Backpropagation Algorithm Works*. Available online: `http://neuralnetworksanddeeplearning.com/chap2.html` (accessed 10 Nov 2025).

[32] Avinash Dixit. *Notes on Optimization*. Princeton University. Available online: `https://www.princeton.edu/~dixitak/Teaching/MicroHighCalculus/Notes&Slides/Optimization.pdf` (accessed 10 Nov 2025).

[33] Stanford University. *Notes on Convexity*. Available online: `https://ai.stanford.edu/~gwthomas/notes/convexity.pdf` (accessed 10 Nov 2025).

[34] Stanford University. *First-Order and Second-Order Optimality Conditions*. Available online: `https://web.stanford.edu/class/msande312/restricted/OPTconditions.pdf` (accessed 10 Nov 2025).

[35] Gilbert Strang. *Linear Algebra and Its Applications*. Brooks/Cole, 4th edition, 2006.

[36] Matthew Bryan. *Multivariable Taylor Series*. Rose–Hulman Institute of Technology. Available online: `https://www.rose-hulman.edu/~bryan/lottamath/mtaylor.pdf` (accessed 10 Nov 2025).

[37] Daniele Vigo. *Introduction to Neural Networks*. Slide del corso, Università di Bologna, 2024–2025.

[38] Neural Networks — Gradient Descent. *3Blue1Brown*. Available online: `https://www.3blue1brown.com/lessons/gradient-descent` (accessed 10 Nov 2025).

[39] What is Backpropagation Really Doing? *3Blue1Brown*. Available online: `https://www.3blue1brown.com/lessons/backpropagation` (accessed 10 Nov 2025).

[40] Backpropagation Calculus. *3Blue1Brown*. Available online: `https://www.3blue1brown.com/lessons/backpropagation-calculus` (accessed 10 Nov 2025).