Fondamenti di Machine Learning

Laurea Triennale in Ingegneria delle Comunicazioni

2: Probabilità, algebra lineare, ottimizzazione

Lecturer: S. Scardapane

SAPIENZA
UNIVERSITÀ DI ROMA

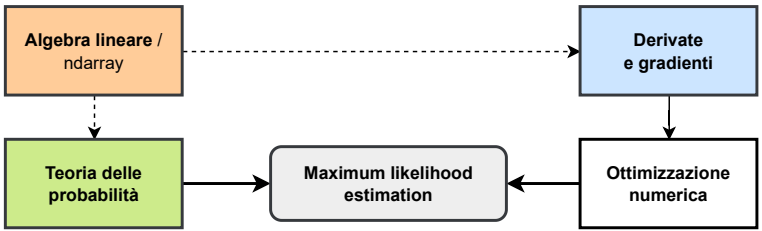


Figure 1: Schema delle varie sezioni della lezione.

Algebra lineare

Array in NumPy

In NumPy, un **ndarray** (o array) è una collezione di oggetti dello *stesso tipo* (es., interi, floating-point, caratteri). Ciascun elemento dell'array è indicizzabile con n valori.

Un array X è descritto da:

- n (a volte chiamato impropriamente *rank*, o *rango*);
- il tipo dei suoi elementi (es., interi);
- il suo **shape**, es., $(4, 5, 2)$ è un array contenente 40 valori.

Ci sono vari modi di inizializzare un array, es.:

```
1 import numpy as np
2 X = np.asarray([2, 1]) # Da una lista o altro oggetto Python
3 X = np.random.randn(4, 5, 2) # `Random' 3-dimensional array
4 X = np.zeros((3, 3)) # Matrice di soli zero
```

Nel terzo caso, la dimensione viene passata come una tupla di valori (si notino le due parentesi consecutive).

Estrarre singoli valori (o sottoinsiemi dell'array) viene detto **indicizzare** l'array. Alcuni esempi:

```
X = np.random.randn(4, 5, 2)
X[0, 3, 1] # Indicizzazione completa (da zero)
X[0]  # Indicizzazione parziale (slice), di dimensioni (5, 2)
X[0:2, 0] # Indicizzazione parziale, di dimensioni (2, 2)
# ...
```

Per quanto riguarda le slide, useremo $X_{i,j,k}$ o X_{ijk} o $[X]_{i,j,k}$ per indicizzare un array, usando le stesse convenzioni che in NumPy.

Numerose operazioni in NumPy si applicano *element-wise*, ad ogni elemento separatamente:

```
X = np.random.randn(4, 5)
Y = np.random.randn(4, 5)
Z = X + np.sin(Y) # Zij = Xij + sin(Yij)
```

A volte scriviamo operazioni che appaiono inconsistenti:

```
X = np.random.randn(4, 5)
y = np.random.randn(5)
Z = X + y # Output shape: (4, 5)
```

Questo va interpretato come $Z_i = X_i + y$ (**broadcasting**).

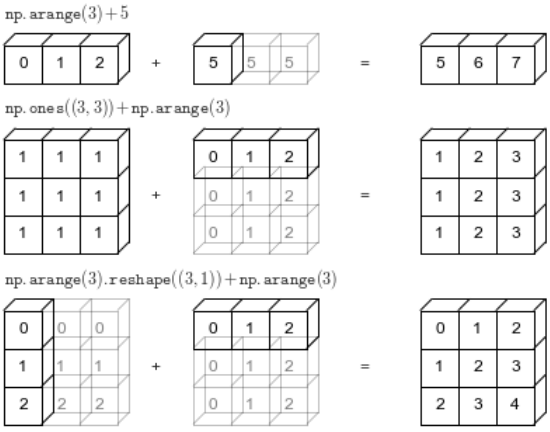


Figure 2: Esempi di broadcasting in NumPy.

Consideriamo questa porzione di codice:

```
a = np.random.randn(3)
b = np.random.randn(3)

# Sum of errors squared
e = ((a - b)**2).sum()

# *WRONG* sum of errors squared
e = ((a.reshape((3,1)) - b.reshape((1,3)))**2).sum()
```

A causa del brodcasting, oggetti con dimensioni (3,), (3,1), o (1,3) sono fondamentalmente diversi.

Algebra lineare

Vettori e matrici

Un array 0-dimensionale viene detto uno **scalare**. Un array 1-dimensionale è invece, nella terminologia dell'algebra lineare, un **vettore** (e lo denotiamo in grassetto):

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{bmatrix}, \quad \mathbf{x}^\top = \begin{bmatrix} x_1 & x_2 & \dots & x_m \end{bmatrix}$$

Possiamo interpretare un vettore come un punto nello spazio (o, equivalentemente, come il segmento che collega l'origine a quel punto).

Possiamo sommare tra loro due vettori (regola del parallelogramma):

$$\mathbf{z} = a\mathbf{x} + b\mathbf{y}\,,\ z_i = ax_i + by_i\,.$$

La *lunghezza* del vettore (norma ℓ_2, la distanza dall'origine), è data da:

$$\|\mathbf{x}\|^2 = \sum_i x_i^2\,. \tag{1}$$

Il **prodotto scalare** (dot product) tra due vettori è invece dato da:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_i x_i y_i = \mathbf{x}^\top \mathbf{y}\,.$$

Geometricamente, il prodotto scalare è legato all'angolo θ tra i due vettori (**cosine similarity**):

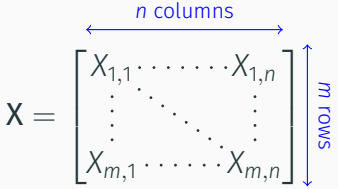
$$\cos(\theta) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|} \,. \tag{2}$$

Per due vettori **ortogonali**, $\langle \mathbf{x}, \mathbf{y} \rangle = 0$. Altrimenti, il prodotto scalare oscilla tra -1 (vettori opposti) e $+1$ (vettori allineati).

La norma Euclidea si può riscrivere in funzione dei prodotti scalari:

$$\|\mathbf{x} - \mathbf{y}\|_2^2 = \langle \mathbf{x}, \mathbf{x} \rangle + \langle \mathbf{y}, \mathbf{y} \rangle - 2\langle \mathbf{x}, \mathbf{y} \rangle \,.$$

Array 2-dimensionali sono detti **matrici**:



$$\mathbf{X} = \begin{bmatrix} X_{1,1} & \cdots & X_{1,n} \\ \vdots & \ddots & \vdots \\ X_{m,1} & \cdots & X_{m,n} \end{bmatrix}$$

Numericamente, una matrice può essere interpretata come un **batch** (insieme) di vettori:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_m \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} \mathbf{x}_{:,1} & \dots & \mathbf{x}_{:,n} \end{bmatrix}$$

Possiamo combinare linearmente due matrici: $\mathbf{Z} = a\mathbf{X} + b\mathbf{Y}$. Generalizzando il prodotto scalare, possiamo anche definire una moltiplicazione tra matrici.

Il **prodotto matriciale** $\underset{(a,c)}{\mathbf{Z}} = \underset{(a,b)}{\mathbf{X}}\,\underset{(b,c)}{\mathbf{Y}}$ è definito come:

$$Z_{i,j} = \langle \mathbf{X}_i, \mathbf{Y}_{:,j} \rangle = \sum_z X_{iz} Y_{zj} \in \mathbb{R}^{a \times c}.$$

Il prodotto tra una matrice ed un vettore rappresenta una proiezione tra i due spazi:

$$\underset{(m)}{\mathbf{b}} = \underset{(m,n)}{\mathbf{W}}\,\underset{(n)}{\mathbf{a}}.$$

Se consideriamo una matrice come un batch di vettori, abbiamo che:

$$\mathbf{X}\mathbf{W} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_m \end{bmatrix} \mathbf{W} = \begin{bmatrix} \mathbf{X}_1\mathbf{W} \\ \vdots \\ \mathbf{X}_m\mathbf{W} \end{bmatrix} \tag{3}$$

Una singola moltiplicazione tra matrici è equivalente ad m prodotti vettori-matrici. Questo tipo di operazioni sono molto importanti in quanto le operazioni in NumPy sono estremamente ottimizzate rispetto ad, es., un for-loop.

Un altro esempio: $\mathbf{X}\mathbf{X}^\top$ è una matrice contenente tutti i prodotti scalari $\langle \mathbf{X}_i, \mathbf{X}_j \rangle$.

Teoria delle probabilità

Regole base della probabilità

Consideriamo questo esempio (riprodotto da Bishop, 2006):

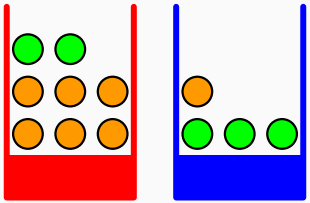


Figure 3: Abbiamo due scatole con diverse proporzioni di palline **arancioni** e **verdi**, provenienti da due scatole (**rossa** e **blu**). Possiamo chiederci "*Qual è la probabilità di estrarre una pallina verde?*", oppure "*Qual è la probabilità di aver estratto dalla prima scatola, dato che il colore della pallina è arancione?*".

Definiamo una variabile C che può prendere i valori *verde* ed *arancione*. La probabilità che una pallina estratta sia di un dato colore è data da:

$$p(C = \text{'verde'}) = 5/12 \tag{4}$$
$$p(C = \text{'arancione'}) = 7/12 \tag{5}$$

$p(C)$ definisce una *distribuzione di probabilità* sui valori di C, e rispetta le seguenti proprietà:

$$p(C = c) \geq 0, \ \sum_{c} p(C = c) = 1 \, .$$

Questa viene detta l'intepretazione *frequentista* della probabilità.

Ora consideriamo una seconda variabile, B, che descrive la scatola (*blu* o *rossa*). Una **distribuzione di probabilità congiunta** coinvolge le due variabili insieme:

$$p(C = \text{‘verde’}, B = \text{‘blu’}) = 2/12.$$

La *regola della somma* delle probabilità ci permette di *marginalizzare* (rimuovere) una delle due variabili:

$$p(C = c) = \sum_b p(C = c, B = b)\,.$$

Ad esempio, $p(C = \text{‘verde’}) = p(C = \text{‘verde’}, B = \text{‘rossa’}) + p(C = \text{‘verde’}, B = \text{‘blu’}) = 2/12 + 3/12 = 5/12$.

La **distribuzione di probabilità condizionale** $p(C \mid B)$ definisce la probabilità di osservare una variabile aleatoria C se conosciamo il valore di un'altra variabile B. Ad esempio (sempre contando le palline):

$$p(C = \text{'verde'} \mid B = \text{'rossa'}) = 2/8.$$

La *regola del prodotto* delle probabilità mette in relazioni le probabilità congiunte e condizionali:

$$p(C = c, B = b) = p(B = b)P(C = c \mid B = b) = p(C = c)p(B = b \mid C = c)\,.$$

Possiamo semplificare la notazione evitando di scrivere esplicitamente i valori delle variabili. Ricapitolando tutti gli assiomi fondamentali di una distribuzione di probabilità:

$$p(C) \geq 0 \qquad \texttt{[Una probabilità è sempre positiva]}$$

$$\sum_{C} p(C) = 1 \qquad \texttt{[La somma delle probabilità è 1]}$$

$$p(C) = \sum_{B} p(C, B) \qquad \texttt{[Regola della somma]}$$

$$p(C, B) = p(B)P(C \mid B) \qquad \texttt{[Regola del prodotto]}$$

Se $p(C, B) = p(C)p(B)$, C e B sono dette **indipendenti**.

Combinando invece la regola della somma e del prodotto, otteniamo la regola di Bayes:

$$p(B \mid C) = \frac{p(C \mid B)p(B)}{p(C)}.$$

Dove $p(C) = \sum_B p(C \mid B)p(B)$. Possiamo interpretare questa regola in vari modi: tra le altre cose, ci permette di aggiornare la nostra conoscenza di B data da $p(B)$ dopo aver osservato un evento C.

Teoria delle probabilità

Distribuzioni di probabilità continue

Consideriamo ora il caso di una variabile x che può assumere valori continui (es., l'altezza di una persona). In questo caso, possiamo definire una **funzione di densità di probabilità** (**probability density function**) $p(x)$ definita come:

$$p(x \in [a, b]) = \int_a^b p(x)dx\,. \tag{6}$$

Tale funzione deve rispettare:

$$p(x) \geq 0, \quad \int p(x)dx = 1\,.$$

In particolare, la **distribuzione di probabilità cumulativa** (**cumulative probability distribution**) è data da:

$$P(x) = \int_{-\infty}^{x} p(y)d(y) \tag{7}$$

da cui $p(x) = P'(x)$. Riotteniamo le regole viste prima in maniera simile:

$$p(x) = \int p(x \mid y)p(y)dy \tag{8}$$

$$p(x, y) = p(x \mid y)p(y) \tag{9}$$

$$p(y \mid x) = \frac{p(x \mid y)p(y)}{\int_y p(x \mid y)p(y)dy} \tag{10}$$

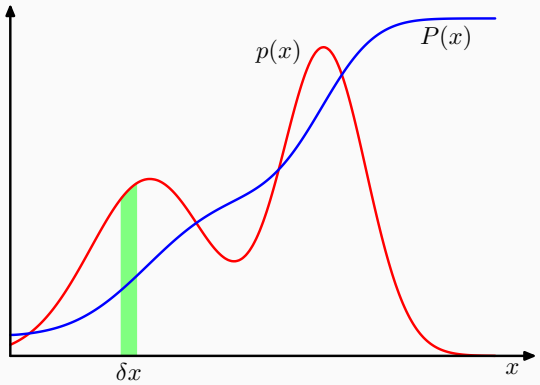


Figure 4: Riprodotta da (Bishop e Bishop, 2023).

Teoria delle probabilità

Manipolare le probabilià

Il **valore atteso** di una funzione $f(x)$ è dato da:

$$\mathbb{E}[f(x)] = \sum_x f(x)p(x)\,.$$

In particolare, la **media** è il valore atteso di $f(x) = x$:

$$\mathbb{E}[x] = \sum_x xp(x)$$

Ad esempio, se vinciamo $x = 1$ per palline verdi e $x = 2$ per palline arancioni, la vincita media è data da $\mathbb{E}[x] = 5/12 + 2 * 7/12 \approx 1.6$.

La **varianza** di x è invece definita come lo spostamento quadratico medio rispetto alla media:

$$\text{Var}(x) = \mathbb{E}\left[(x - \mathbb{E}[x])^2\right] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 \tag{11}$$

Ritornando all'esempio delle palline, la varianza ci permette di calcolare quanto potrebbero variare le nostre vincite individuali su varie ripetizioni.

Supponiamo di non avere accesso a $p(x)$ (non possiamo guardare dentro le scatole), ma possiamo campionare valori $x \sim p(x)$ (estrarre una pallina). Con un sufficiente numero di campioni possiamo stimare il valore atteso:

$$\mathbb{E}[f(x)] = \sum_x f(x)p(x) \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i) \tag{12}$$

dove $x_1, \ldots, x_n$ sono i campioni. Questo viene detto **metodo di Monte Carlo**.

Teoria delle probabilità

Distribuzioni di probabilità parametriche

Un concetto che useremo ripetutamente è quello di **one-hot encoding** (o **one-of-k encoding**, o **dummy encoding**).

Consideriamo una variabile discreta $x \in \{1, \ldots, k\}$. Il suo one-hot encoding è un vettore $\mathbf{x} \in \{0, 1\}^k$ tale che:

$$x_i = \begin{cases} 1 & \text{se } x = i \\ 0 & \text{altrimenti} \end{cases} \tag{13}$$

Consideriamo $x \in \{\text{`}verde\text{'}, \text{`}blu\text{'}, \text{`}arancione\text{'}\}$. L'encoding diventa:

$$\text{`}verde\text{'} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \text{`}blu\text{'} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \text{`}arancione\text{'} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{14}$$

Il dummy encoding permette di trasformare una variabile categorica in un vettore, senza imporre un ordinamento tra gli elementi.

Raggruppiamo le probabilità associate ad x in un vettore $\mathbf{p} = [p_1, \dots, p_k]^\top$. Possiamo descrivere l'intera distribuzione di probabilità in maniera concisa come:

$$\text{Cat}(\mathbf{x}; \mathbf{p}) = \prod_{i=1}^{k} p_i^{x_i} = p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k}, \tag{15}$$

dove $\mathbf{x}$ è il dummy encoding di x. Scritta in questa forma, viene detta una **distribuzione categorica**. La notazione $\text{Cat}(x; \mathbf{p})$ chiarisce che $\mathbf{p}$ sono gli unici parametri necessari a specificare interamente la distribuzione stessa: vengono detti **sufficient statistic**.

Nel caso di variabili aleatorie continue, una distribuzione molto comune è quella **Gaussiana**:

$$p(x) = \mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \tag{16}$$

Questa rappresenta una distribuzione centrata in μ (la **media**), che decresce sui lati in maniera simmetrica con uno spread che dipende da σ^2 (la **varianza** della distribuzione).

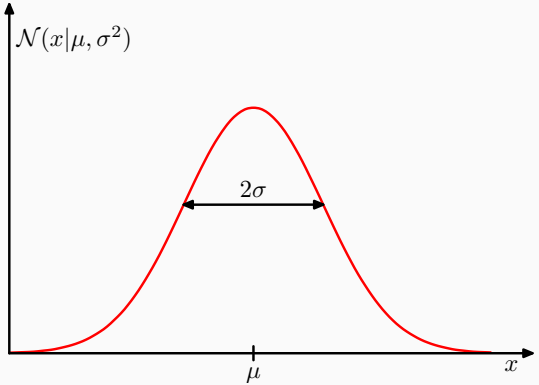


Figure 5: Riprodotto da (Bishop e Bishop, 2023).

Teoria delle probabilità

Maximum likelihood

Supponiamo di non conoscere $p(x)$, ma di avere accesso ad n campioni $\{x_i\}, x_i \sim p(x)$.

Abbiamo però l'intuizione che la distribuzione abbia una forma $f(x; s)$, di cui non conosciamo però le statistiche sufficienti s. Il principio del **maximum likelihood** (massima verosimiglianza) ci permette di stimare questi valori, se assumiamo che essi siano stati campionati in maniera indipendente l'uno dall'altro.

Essendo i campioni indipendenti, possiamo scrivere una distribuzione congiunta fattorizzata:

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} f(x_i; s) \tag{17}$$

Banalmente, i parametri ottimi sono quelli che massimizzano questa quantità:

$$s^* = \arg\max_s \prod_{i=1}^{n} f(x_i; s) \tag{18}$$

Se n è grande, produttorie di tantissimi elementi possono essere instabili a livello numerico. In questo caso, possiamo definire la **log-likelihood** come il logaritmo della likelihood:

$$\log p(x_1, \ldots, x_n) = \sum_{i=1}^{n} \log f(x_i; s) \tag{19}$$

Poiché il logaritmo è monotono, massimizzare la log-likelihood è equivalente a massimizzare la likelihood:

$$s^* = \arg\max_s \sum_{i=1}^{n} \log f(x_i; s) \tag{20}$$

Vedremo più avanti che possiamo risolvere problemi di ottimizzazione di questo tipo tramite **discesa al gradiente**. Consideriamo prima due esempi semplici di stima ML (per le derivazioni si veda (Bishop e Bishop, 2023)).[1]

Consideriamo una distribuzione categorica. Otteniamo:

$$p_i = \frac{\sum_{j=1}^{n} [x_i]_j}{n} \tag{21}$$

che non è altro che la frazione di valori i osservati tra gli n campionati.

Consideriamo ora una Gaussiana. Possiamo riscrivere la log-likelihood come:

$$L(\mu, \sigma^2) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{22}$$

Da cui otteniamo (utile esercizio provarci):

$$\mu^* = \frac{1}{n}\sum_i x_i \tag{23}$$

$$\sigma^{2*} = \frac{1}{n}\sum_i (x_i - \mu^*)^2 \tag{24}$$

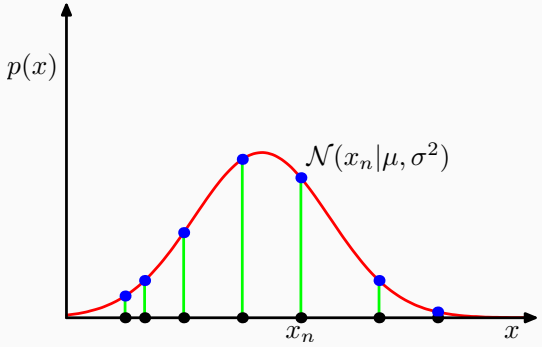


Figure 6: Riprodotta da (Bishop e Bishop, 2023). Trovare la media ottima (secondo il principio della massima verosimiglianza) vuol dire scegliere μ tale da minimizzare il quadrato della somma dei segmenti verdi.

Ottimizzazione numerica

Derivate e gradienti

Consideriamo una funzione $f(x) : \mathbb{R} \to \mathbb{R}$. Sotto certe condizioni, possiamo studiarne l'andamento tramite la sua derivata.

La **derivata** di una funzione $f(x)$ è definita come:

$$\partial f(x) = f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{25}$$

Anche per una funzione continua, non è detto che la derivata sia definita ovunque.

La derivata descrive come varia $f(x)$ per uno spostamento 'infinitesimo' rispetto ad x.

Derivata di un polinomio:

$$\partial [x^p] = p x^{p-1} .$$

Derivata di funzioni esponenziali e logaritmi:

$$\partial [\exp(x)] = \exp(x) ,$$
$$\partial [\log(x)] = \frac{1}{x} .$$

Le derivate possiedono numerose proprietà algebriche, tra cui:

- **Linearità**:
$$\partial\Big[f(x) + g(x)\Big] = f'(x) + g'(x)\,.$$
- **Regola del prodotto**:
$$\partial\Big[f(x)g(x)\Big] = f'(x)g(x) + f(x)g'(x)\,,$$
- **Chain rule (regola della composizione)**
$$\partial\Big[f(g(x))\Big] = f'(g(x))g'(x)\,.$$

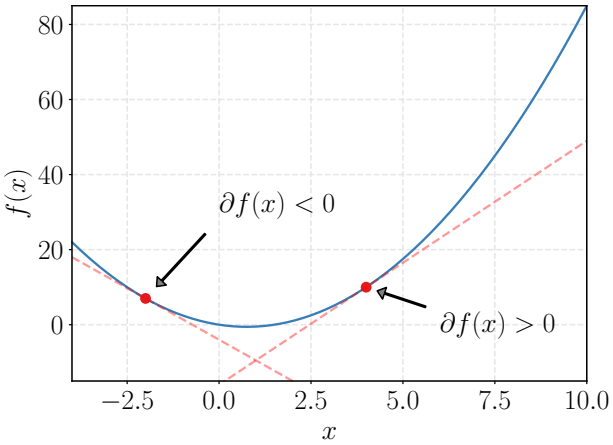


Figure 7: $f(x) = x^2 - 1.5x$. La derivata in ogni punto può essere interpretata come la pendenza della linea tangente in quel punto.

Per una funzione $y = f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^m$, il **gradiente** $\partial f(\mathbf{x})$ è un vettore m-dimensionale:

$$[\partial f(\mathbf{x})]_i = \frac{\partial y}{\partial x_i} = \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h}, \tag{26}$$

dove $\mathbf{e}_i$ è il vettore:

$$[\mathbf{e}_i]_j = \begin{cases} 1 & \text{se } i = j \\ 0 & \text{altrimenti} \end{cases}$$

A volte useremo la notazione $\nabla f(\mathbf{x})$ per indicare il gradiente.

Più in generale, la **derivata direzionale** di $f(\mathbf{x})$ lungo una direzione $\mathbf{v}$ è data da:

$$\mathrm{D}_{\mathbf{v}} f(\mathbf{x}) = \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{v}) - f(\mathbf{x})}{h}, \tag{27}$$

Si può provare la seguente identità (informalmente, il movimento si decompone rispetto agli assi):

$$\mathrm{D}_{\mathbf{v}} f(\mathbf{x}) = \langle \nabla f(\mathbf{x}), \mathbf{v} \rangle . \tag{28}$$

Nel caso più generale $\mathbf{y} = f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^n$:

Lo **Jacobian** $\underset{(n,m)}{\partial f(\mathbf{x})}$ di f è definito come:

$$\partial f(\mathbf{x}) = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_m} \end{pmatrix}. \tag{29}$$

Per $n = 1$ otteniamo il gradiente, per $m = n = 1$ otteniamo la derivata.

Prodotto scalare:

$$\frac{\partial}{\partial \mathbf{x}} \langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{y} \,.$$

Moltiplicazione di un vettore e di una matrice:

$$\frac{\partial}{\partial \mathbf{x}} \mathbf{A}\mathbf{x} = \mathbf{A} \,.$$

Gradiente della norma:

$$\partial \|\mathbf{x}\|^2 = 2\mathbf{x} \,.$$

Si veda come riferimento:
`https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf`.

Quasi tutte le regole delle derivate si estendono al caso multi-variato, ad esempio (linearità):

$$\partial(f(\mathbf{x}) + g(\mathbf{x})) = \partial f(\mathbf{x}) + \partial g(\mathbf{x}) \tag{30}$$

Esiste anche una chain rule: date $f : \mathbb{R}^m \to \mathbb{R}^n$ e $g : \mathbb{R}^o \to \mathbb{R}^m$, con $\mathbf{h} = g(\mathbf{x})$:

$$\underset{(n,o)}{\partial\,[f(g(\mathbf{x}))]} = \underset{(n,m)}{\partial f(\mathbf{h})} \underset{(m,o)}{\partial g(\mathbf{x})}\,. \tag{31}$$

In parole: lo Jacobiano della composizione è il prodotto (matriciale) dei due Jacobiani individuali.

Data una funzione $f(\mathbf{x}_0)$ valutata in $\mathbf{x}_0$, la funzione:

$$\widetilde{f}(\mathbf{x}) = f(\mathbf{x}_0) + \langle \partial f(\mathbf{x}_0), \mathbf{x} - \mathbf{x}_0 \rangle \tag{32}$$

è la miglior **approssimazione lineare** di f rispetto a $\mathbf{x}_0$ (teorema di Taylor). Considerando derivate di ordini superiori è possibile migliorare la qualità dell'approssimazione.

```python
# Function
f = lambda x: x**2 - 1.5*x

# Derivative (manual)
df = lambda x: 2*x - 1.5

# Linearization at 0.5
x = 0.5
f_linearized = lambda h: f(x) + df(x)*(h - x)

print(f(x + 0.01)) # -0.5049
print(f_linearized(x + 0.01)) # -0.5050
```

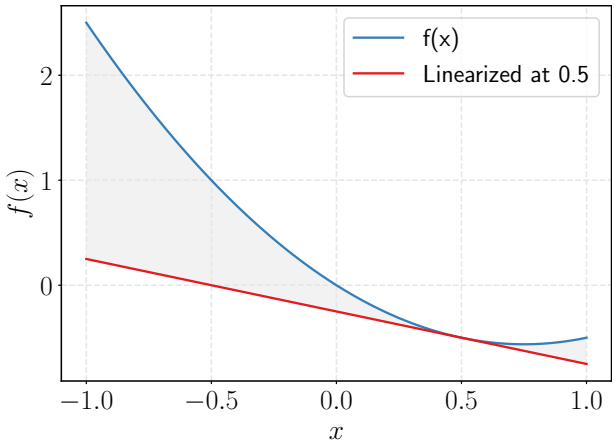


Figure 8: Funzione ($f(x) = x^2 - 1.5x$), linearizzata in 0.5.

Ottimizzazione numerica

Discesa al gradiente

Consideriamo nuovamente un problema di questo tipo:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}) \tag{33}$$

Si tratta di un problema di **ottimizzazione (non vincolata)** nel dominio $\mathbb{R}^d$. Massimizzare rispetto a minimizzare è una scelta stilistica, in quanto:

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x}) = \arg\min_{\mathbf{x} \in \mathbb{R}^d} -f(\mathbf{x}) \tag{34}$$

Il maximum likelihood è un esempio di tale problema (se la likelihood è differenziabile).

Un punto $\mathbf{x}$ per il quale $f(\mathbf{x}) \leq f(\mathbf{x}') \ \ \forall\, \mathbf{x}' \in \mathbb{R}^d$ viene detto un **minimo globale**. In maniera meno restrittiva, invece, se:

$$f(\mathbf{x}) \leq f(\mathbf{x}') \ \ \forall\, \mathbf{x}' \in \left\{\mathbf{x}' : \|\mathbf{x}' - \mathbf{x}\|^2 < \varepsilon\right\} \tag{35}$$

per qualche $\varepsilon > 0$, allora il punto viene detto un **minimo locale**.

Se $\nabla f(\mathbf{x}) = 0$, $\mathbf{x}$ il punto viene detto **stazionario**. Come vedremo, i punti stazionari possono essere minimi, massimi, oppure **punti di sella**.

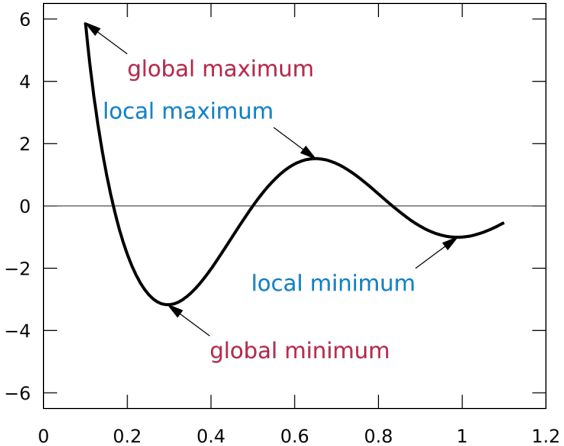


Figure 9: Senza nessuna informazione aggiuntiva, un punto stazionario può essere un **minimo**, un **massimo**, e può essere **locale** o **globale** (*Wikimedia, KSmrq*).

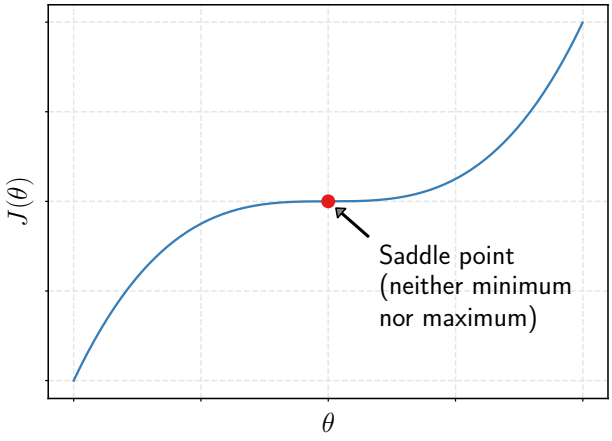


Figure 10: I punti stazionari possono essere anche **punti di sella**, che crescono e decrescono in direzioni diverse.

Consideriamo un punto (scelto casualmente) $\mathbf{x}_0$, ed un algoritmo iterativo in cui ad ogni istante ci muoviamo in una direzione $\mathbf{p}_t$ per una lunghezza η_t:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \eta_t \mathbf{p}_t \,. \tag{36}$$

$\mathbf{p}_t$ è chiamata una **descent direction** in $f(\mathbf{x}_{t-1})$ se $f(\mathbf{x}_t) < f(\mathbf{x}_{t-1})$ per qualche η_t, anche molto piccolo. η_t viene detto **step size** o **learning rate**.

Poiché $\mathbf{p}_t$ descrive solo la direzione del movimento, possiamo assumere che $\|\mathbf{p}_t\| = 1$. Possiamo quantificare il cambiamento (infinitesimo) dello spostamento con la derivata direzionale:

$$\mathrm{D}_{\mathbf{p}_t} f(\mathbf{x}_{t-1}) = \langle \nabla f(\mathbf{x}_{t-1}), \mathbf{p}_t \rangle = \|\nabla f(\mathbf{x}_{t-1})\| \underbrace{\|\mathbf{p}_t\|}_{=1} \cos(\theta) = \|\nabla f(\mathbf{x}_{t-1})\| \cos(\theta).$$

Dall'equazione si vede che la discesa è massima quando $\cos(\theta) = -1$, cioè quando $\theta = \pi$ e $\mathbf{p}_t = -\nabla f(\mathbf{x}_{t-1})$. Questa viene chiamata la **steepest descent direction**. In generale, qualsiasi direzione tale per cui $\cos(\theta) < 0$ è una descent direction.

Un algoritmo iterativo in cui scegliamo sempre di muoverci nella steepest descent direction viene detto **discesa al gradiente**.

La discesa al gradiente trova un punto stazionario di una funzione $f(\cdot)$ tramite la seguente procedura iterativa:

$$\mathbf{x}_t = \mathbf{x}_{t-1} - \eta_t \nabla f(\mathbf{x}_{t-1}) \,. \tag{37}$$

Il concetto di **convessità** è essenziale nell'ottimizzazione. Se una funzione è convessa, la sua ottimizzazione è più semplice in quanto non esistono minimi locali.

f è convessa se per $\lambda \in [0, 1]$:

$$f\Big((1-\lambda)\mathbf{x}_1 + \lambda\mathbf{x}_2\Big) \leq (1-\lambda)f(\mathbf{x}_1) + \lambda f(\mathbf{x}_2)\,. \tag{38}$$

Se questo vale sempre in maniera stretta, f viene detta **strettamente convessa** (strictly convex).

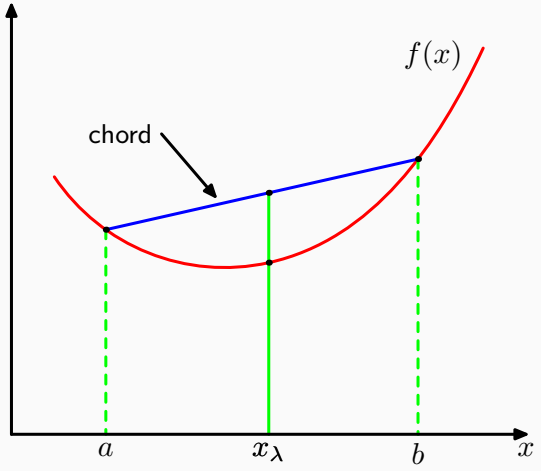


Figure 11: Riprodotto da (Bishop e Bishop, 2023).

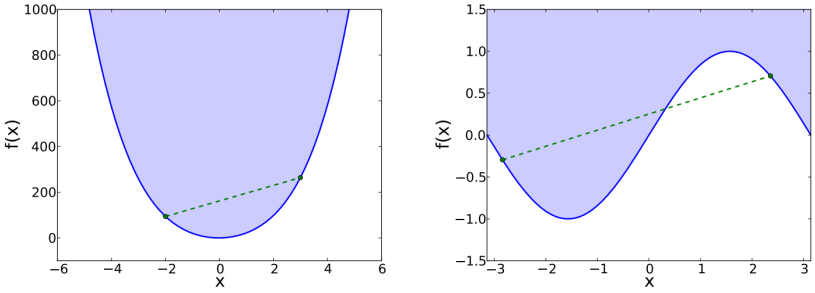


Figure 12: Sinistra: un esempio di funzione convessa. Destra: un esempio di funzione non convessa. Riprodotto da “An Introduction to Machine Learning” by Smola and Vishwanathan [unpublished].

Consideriamo una funzione $f(\mathbf{x})$, e supponiamo la discesa al gradiente converga ad un punto $\mathbf{x}^*$. Allora:

- Se $f(\mathbf{x})$ è non convessa: il punto è **stazionario**.
- Se $f(\mathbf{x})$ è convessa: il punto è un **minimo globale**.
- Se $f(\mathbf{x})$ è strettamente convessa: il punto è **l'unico** minimo globale.

Per una funzione non convessa questo risultato non può essere migliorato (informalmente: l'unico modo di trovare un ottimo globale è inizializzare la discesa al gradiente da *qualsiasi* punto dello spazio).