

Fondamenti di Machine Learning

Laurea Triennale in Ingegneria delle Comunicazioni

3: Apprendimento supervisionato e regressione lineare

Lecturer: S. Scardapane



SAPIENZA
UNIVERSITÀ DI ROMA

Apprendimento supervisionato

Definizioni di base

L'**apprendimento supervisionato** ha l'obiettivo di inferire una relazione tra due classi di oggetti, l'**input space** \mathcal{X} e l'**output space** \mathcal{Y} , sulla base di dati storici.

Ad esempio, un elemento $x \in \mathcal{X}$ può essere una rappresentazione di una mail (es., lista ordinata di parole), mentre il corrispondente output \mathcal{Y} può essere 0 per una mail valida, 1 per una mail di spam.

- ▶ **Classificazione:** gli elementi di \mathcal{Y} sono in numero finito, $\{1, \dots, M\}$. Se $M = 2$, come nel caso delle email, parliamo di **classificazione binaria**. Altrimenti, parliamo di **classificazione multi-classe**.
- ▶ **Regressione:** \mathcal{Y} è un insieme infinito, es., la temperatura di una stanza. In un problema di regressione vogliamo predire una informazione *quantitativa* invece che *qualitativa*.

1. **Speech recognition:** x è un recording audio, e y la sua trascrizione.
2. **Robot navigation:** x è l'output di due telecamere montate su un robot, e y il comando da eseguire.
3. **Text translation:** x è una frase in inglese, e y la sua traduzione in francese.
4. **Product recommendation:** x è un utente, e y la sua affinità ad un certo prodotto nel catalogo.

In questo corso assumiamo che l'input sia un vettore di numeri reali, che scriviamo in grassetto \mathbf{x} , ed abbiamo quindi $\mathcal{X} = \mathbb{R}^d$. Chiamiamo \mathbf{x} un **pattern**.

Molti oggetti di interesse possono essere ridotti in questa forma tramite una qualche trasformazione. Ad esempio, un testo può essere convertito in un vettore che conta la presenza di un certo insieme di parole (**bag-of-words**).

Un singolo elemento x_i viene detto una **feature** nel machine learning, o una **variabile indipendente** in statistica.

Una classe importante di feature sono quelle dette **categoriche**. Ad esempio, in una applicazione medica una feature può distinguere il colore degli occhi di un paziente. Come vedremo, rappresentarla con $\{0, 1, 2, \dots\}$ è problematico perché impone un ordinamento sulle categorie.

Se il numero dei possibili valori è basso, è possibile usare un **dummy encoding** (o 1-of-K encoding, o **one-hot encoding**), dove usiamo un singolo bit per ogni possibile valore:

$$\text{Verdi} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \text{Marroni} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \text{Blu} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Apprendimento supervisionato

Formalizzazione del problema

Un dataset (etichettato, *supervised*) è un insieme di n **esempi** della relazione desiderata:

$$\mathcal{S} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\} . \quad (1)$$

Informalmente, dato un ‘nuovo’ esempio (\mathbf{x}, y) non contenuto in \mathcal{S} , vorremmo una funzione $f(\cdot)$ tale che:

$$f(\mathbf{x}) \approx y . \quad (2)$$

Più in generale, possiamo testare il modello su un secondo dataset \mathcal{T} indipendente, i.e., $\mathcal{S} \cap \mathcal{T} = \emptyset$, detto **test set**.

In generale, non possiamo assumere che y_i sia determinato *univocamente* da x_i per diverse ragioni:

- ▶ Potrebbero esistere altre feature che determinano y_i che non osserviamo (**latent variables**).
- ▶ Alcune feature di x_i potrebbero mancare od essere errate, così come i valori di y_i potrebbero essere corrotti da rumore (es., sensori).
- ▶ Potrebbe comunque essere impossibile effettuare una previsione unica (es., predire l'evoluzione del prezzo di una azione nei prossimi sei mesi).

Date queste condizioni di incertezza, possiamo assumere genericamente che i nostri dati sono descritti da una *distribuzione di probabilità*:

$$p(\mathbf{x}_i, y_i) = p(\mathbf{x}_i)p(y_i \mid \mathbf{x}_i) = p(y_i)p(\mathbf{x}_i \mid y_i).$$

In questa formalizzazione, un dataset diventa una variabile aleatoria data dal campionamento di n valori indipendenti e identicamente distribuiti (i.i.d.). \mathcal{S} e \mathcal{T} sono quindi dati da due campionamenti separati.

$$p(\mathcal{S}) = \underbrace{\prod_{i=1}^n p(\mathbf{x}_i, y_i)}_{\text{Likelihood}}$$

$$\log p(\mathcal{S}) = \underbrace{\sum_{i=1}^n \log p(\mathbf{x}_i, y_i)}_{\text{Log-likelihood}}$$

L'assunzione che gli elementi del training set \mathcal{S} e del test set \mathcal{T} siano campionati i.i.d. dalla stessa distribuzione è importante ma restrittiva:

- ▶ **Identicamente distribuiti:** la distribuzione $p(\cdot, \cdot)$ è identica e non varia (es., nel riconoscere specie di gatti, esse sono stabili nel tempo).
- ▶ **Indipendenti:** non esiste un bias nel modo in cui campioniamo i dati (es., collezionare foto di gatti in un unico quartiere).

Gli algoritmi di apprendimento supervisionato si differenziano a seconda di quale termine cercano di approssimare:

1. Algoritmi che approssimano $p(x | y)$ o $p(x)$ sono detti **generativi** (es., **linear discriminant analysis**).
2. Algoritmi che approssimano solo $p(y | x)$ vengono detti **discriminativi**.

Nella maggior parte del corso ci concentreremo su approcci discriminativi, in quanto più semplici concettualmente.

“When solving a problem of interest, do not solve a more general problem as an intermediate step.”

— V. Vapnik, Statistical learning theory, 1998

Supponiamo di essere interessati solo alla classe più probabile, o al valore più probabile nel caso di regressione. In questo caso:

$$y^* = \arg \max_y p(y | \mathbf{x})p(\mathbf{x}) = \arg \max_y \log(p(y | \mathbf{x})) + \log(p(\mathbf{x})) \quad (3)$$

$$\arg \max_y \log(p(y | \mathbf{x})) + \log(p(\mathbf{x})) = \arg \max_y \log(p(y | \mathbf{x})). \quad (4)$$

Quindi, la cosa che ci interessa maggiormente è una funzione $f(\mathbf{x})$ che approssimi il termine che massimizza la log-probability della classe.

Model-based supervised learning

Modelli lineari e Least Squares

Il modello più semplice per $f(\mathbf{x})$ è una funzione **lineare**: come vedremo, questo semplifica l'implementazione e la soluzione del problema di ottimizzazione.

Un modello lineare f è definito come:

$$f(\mathbf{x}) = \sum_{i=1}^d w_i x_i + b = \mathbf{w}^\top \mathbf{x} + b , \quad (5)$$

dove \mathbf{w} e b (**bias** o **intercept**) sono parametri del modello.

Nel caso di una sola feature, questa è la classica retta $f(x) = mx + q!$

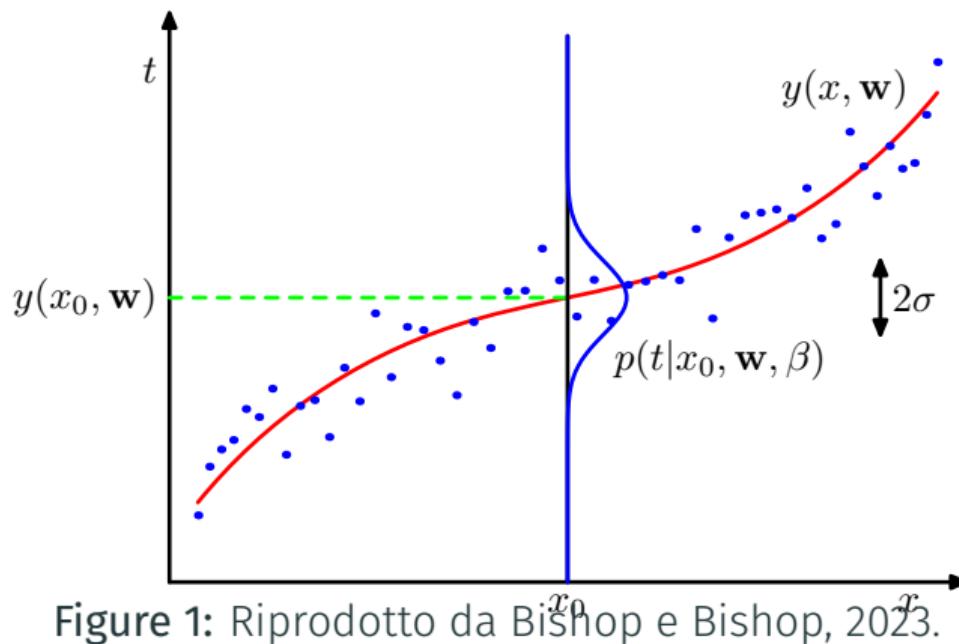


Figure 1: Riprodotto da Bishop e Bishop, 2023.

Supponendo che una feature del nostro input sia sempre costante, $\mathbf{x} = [x_1, \dots, x_{d-1}, 1]$, possiamo semplificare la notazione rimuovendo il bias:

$$f(\mathbf{x}) = \sum_{i=1}^d w_i x_i = \sum_{i=1}^{d-1} w_i x_i + w_d.$$

Nel seguito usiamo questa notazione per evitare di scrivere esplicitamente il bias nelle equazioni, che però non va mai dimenticato a livello implementativo!

Come scegliere i parametri w del modello lineare? Intuitivamente, possiamo cercare i parametri che minimizzano la discrepanza (l'errore) tra le predizioni e i valori reali nel training set.

Una scelta naturale è minimizzare la somma dei quadrati degli errori (**sum of squared residuals**):

$$w^* = \arg \min_w \sum_{i=1}^n (y_i - f(x_i))^2 \quad (6)$$

Il valore ottimale dei parametri w è quello che minimizza la somma dei residui al quadrato: per questa ragione, questo metodo viene detto **least-squares** (LS) regression.

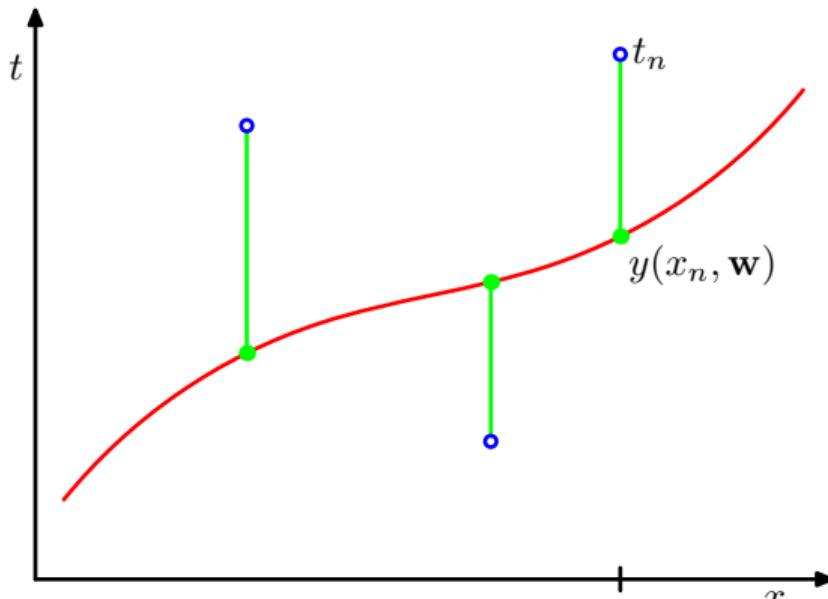


Figure 2: Scegliamo i parametri w che minimizzano la somma degli scostamenti (in verde) al quadrato.

Model-based supervised learning

Interpretazione probabilistica

Possiamo dare una giustificazione formale al criterio least-squares? Per farlo, supponiamo che $f(\mathbf{x})$ predica la media di una Gaussiana, la cui varianza, che modella il rumore sui dati, è fissata a σ^2 :

$$p(y | \mathbf{x}) = \mathcal{N}(y | f(\mathbf{x}), \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{y - f(\mathbf{x})}{\sigma}\right)^2\right) \quad (7)$$

Data questa assunzione, cerchiamo i parametri che massimizzano la probabilità dei dati che abbiamo osservato (il nostro training dataset).

Come detto sopra, questo è il criterio di **maximum likelihood**:

$$\mathbf{w}^* = \arg \max_{\mathbf{w}} \prod_{i=1}^n p(y_i | f(\mathbf{x}_i), \sigma^2) = \arg \max_{\mathbf{w}} \sum_{i=1}^n \log(p(y_i | f(\mathbf{x}_i), \sigma^2)) \quad (8)$$

Vediamo cosa succede espandendo il termine all'interno della sommatoria.

Ricordando il logaritmo della Gaussiana:

$$\log(p(y | \mathbf{w}^\top \mathbf{x}, \sigma^2)) = -\frac{1}{2} \log(\sigma) - \frac{1}{2} \log(2\pi) - \frac{1}{\sigma^2} (y - \mathbf{w}^\top \mathbf{x})^2 \quad (9)$$

Massimizzando per \mathbf{w} vediamo che i primi due termini sono costanti e rimane solo il terzo termine.

Massimizzare la likelihood (con rumore Gaussiano) è equivalente a minimizzare la somma dei quadrati degli errori (LS)!

$$\arg \max_{\mathbf{w}} \sum_{i=1}^n \log(p(y_i | f(\mathbf{x}_i))) \longleftrightarrow \arg \min_{\mathbf{w}} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2 \quad (10)$$

Dato \mathbf{w}^* , possiamo ottimizzare la maximum likelihood anche rispetto a σ^2 , da cui si ottiene la relazione molto intuitiva:

$$\sigma^{2,*} = \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w}^{*,\top} \mathbf{x})^2. \quad (11)$$

In questo caso, l'incertezza del modello è fissa per ogni \mathbf{x} (**modello omoscedastico**), ed è data dalla media dei residui al quadrato.

Un vantaggio dell'approccio probabilistico è che possiamo facilmente rilassare questa assunzione e considerare un modello in cui la varianza dipende dall'input:

$$p(y | \mathbf{x}) = \mathcal{N}(y | f_{\mu}(\mathbf{x}), f_{\sigma^2}(\mathbf{x})). \quad (12)$$

In questo caso (**modello eteroscedastico**) possiamo apprendere sia la media che la varianza dai dati, massimizzando la likelihood congiunta. Questo è molto utile in applicazioni reali dove l'incertezza non è uniforme.

Visualizzazione (modello eteroscedastico)

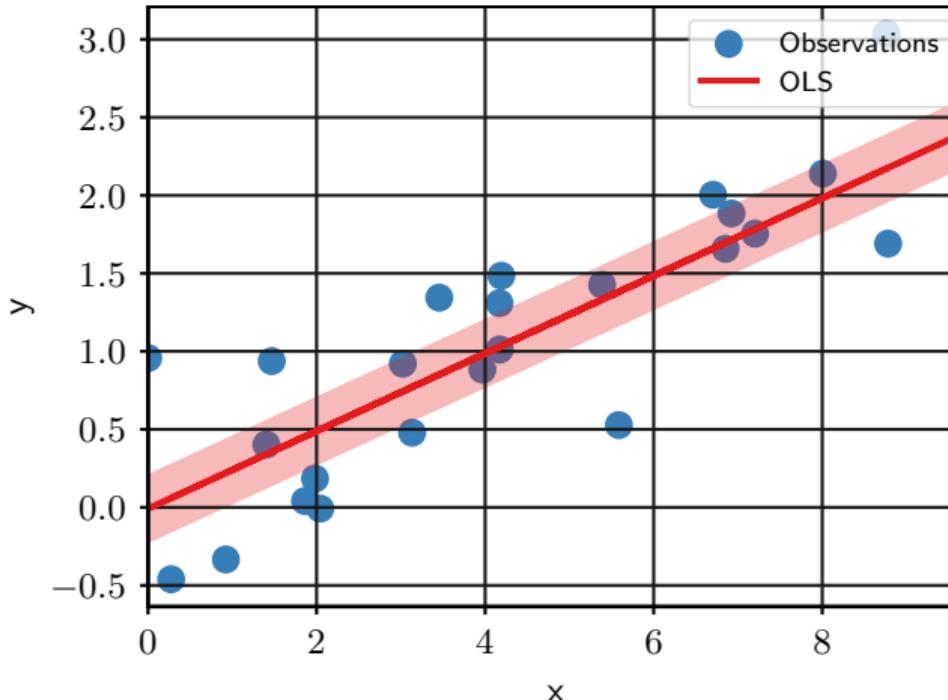


Figure 3: Esempio di regressione con incertezza predittiva appresa dai dati (banda corrispondente a $\pm\sigma(x)$).

Model-based supervised learning

Empirical risk minimization

Abbiamo seguito due strade diverse per arrivare alla stessa idea.

1. **Intuitiva/Geometrica:** minimizzare l'errore.
2. **Probabilistica:** massimizzare la verosimiglianza dei dati osservati assumendo rumore Gaussiano.

I due approcci danno due prospettive diverse ed ugualmente interessanti: ad esempio, ragionare in termini di errore ci permette di definire funzioni di costo alternative, mentre ragionare su p permette di integrare misurare di incertezza nella predizione.

Il termine $(y_i - f(\mathbf{x}_i))^2$ rappresenta l'errore (il *residuo*) che la funzione f ha sull'esempio (\mathbf{x}_i, y_i) . Possiamo generalizzare questa idea introducendo il concetto di **loss function** (o **funzione di costo**).

Una **loss function** $L(y, \hat{y}) \in \mathbb{R}$ associa ad ogni predizione $\hat{y} = f(\mathbf{x})$ uno scalare, tale per cui $L(y, \hat{y}_1) < L(y, \hat{y}_2)$ implica che \hat{y}_1 è una miglior predizione di \hat{y}_2 .

Visualizzazione della squared loss

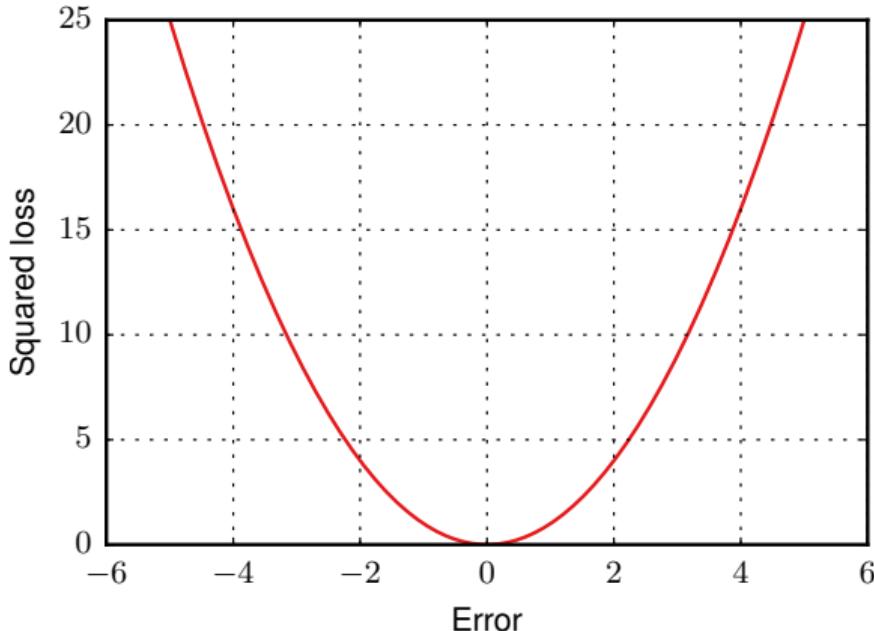


Figure 4: Visualizzazione della squared loss al variare del residuo $e = y - f(x)$.

Da questa seconda prospettiva (**statistical learning**) possiamo direttamente *definire* una funzione di costo L che ci sembri adeguata.

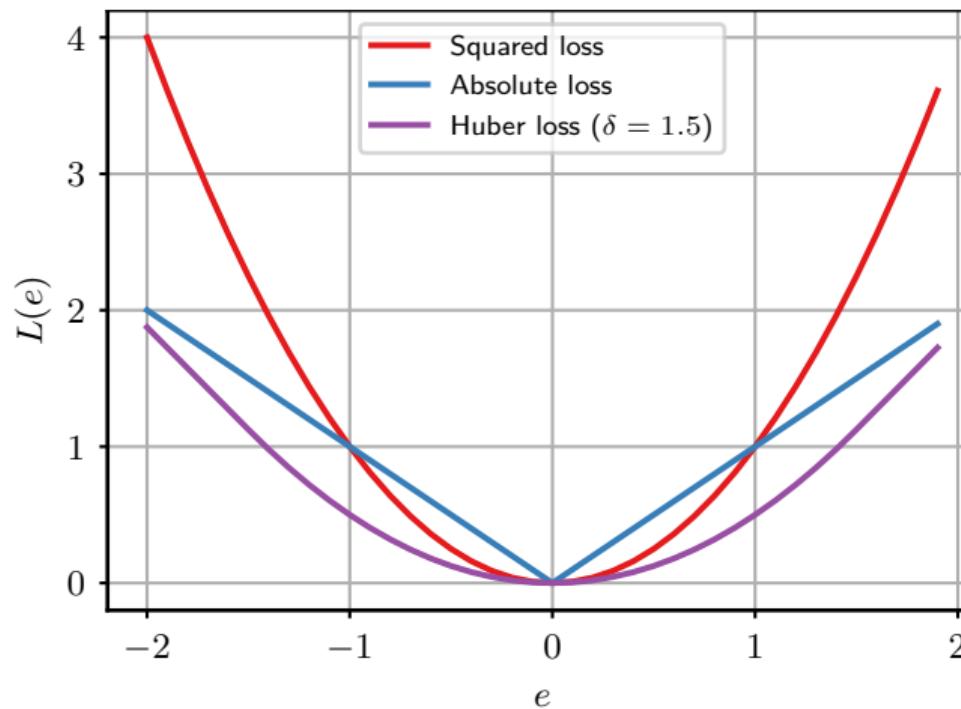
Ad esempio, per evitare di pesare quadraticamente errori sempre più alti, potremmo considerare il valore assoluto:

$$\text{Absolute deviation: } L(y, \hat{y}) = |y - \hat{y}|. \quad (13)$$

Combinando le due loss otteniamo la **Huber loss**, per un $\sigma > 0$ a scelta:

$$\text{Huber loss: } L(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{se } |y - \hat{y}| \leq \sigma \\ \sigma(|y - \hat{y}| - \frac{1}{2}\sigma) & \text{altrimenti} \end{cases} \quad (14)$$

Confronto di varie loss per la regressione



Data una funzione di costo, possiamo definire un problema di apprendimento supervisionato come segue.

Il **rischio atteso** di una funzione $f(\mathbf{x})$ è dato da:

$$f^*(\mathbf{x}) = \arg \min_f \left\{ \mathbb{E}_{p(\mathbf{x}, y)} [L(y, f(\mathbf{x}))] \right\}. \quad (15)$$

Il rischio atteso non è calcolabile (non avendo accesso alla distribuzione), ma dato un dataset può essere approssimato come la sua media empirica (**empirical risk minimization**):

$$f^*(\mathbf{x}) = \arg \min_f \left\{ \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i)) \right\}. \quad (16) \quad 29$$

Per rendere più chiaro il confronto, ricordiamo che:

$$\arg \min_w L(w) = \arg \min_w aL(w) + b \quad (17)$$

per qualsiasi scelta di a e b . Quindi:

$$\arg \min_w \frac{1}{\sigma^2} \sum_{i=1}^n (y_i - f(x_i))^2 = \arg \min_w \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2 \quad (18)$$

Minimizzare la likelihood o minimizzare il rischio empirico (con MSE) porta alla stessa soluzione ottima.

Model-based supervised learning

Soluzione del least-squares

Per semplificare la notazione (ed avvicinarci ad una implementazione reale), definiamo la **matrice di input** ed il **vettore di output** come:

$$\underset{(n,d)}{\mathbf{X}} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \underset{(n)}{\mathbf{y}} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} .$$

Ogni input è una riga della matrice. Il problema del least-squares diventa:

$$\text{LS}(\mathbf{w}) = \frac{1}{n} (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) = \frac{1}{n} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 .$$

Per risolvere il problema calcoliamo il gradiente (anche dette le **equazioni normali**):

$$\nabla \text{LS}(\mathbf{w}) = \frac{2}{n} \mathbf{X}^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) . \quad (19)$$

Partendo da una inizializzazione casuale \mathbf{w}_0 , la discesa al gradiente è quindi:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t \nabla \text{LS}(\mathbf{w}_{t-1}) . \quad (20)$$

Questa sequenza converge all'ottimo globale del least-squares, come si dimostra analizzandone la convessità.

Per iniziare, proviamo che la norma $\|\cdot\|^2$ è strettamente convessa:

$$\|\lambda \mathbf{x}_1 + (1 - \lambda) \mathbf{x}_2\|^2 \leq \lambda \|\mathbf{x}_1\|^2 + (1 - \lambda) \|\mathbf{x}_2\|^2 \quad (21)$$

Espandendo i termini:

$$-\lambda(1 - \lambda)\mathbf{x}_1^\top \mathbf{x}_1 - \lambda(1 - \lambda)\mathbf{x}_2^\top \mathbf{x}_2 + 2\lambda(1 - \lambda)\mathbf{x}_1^\top \mathbf{x}_2 \leq 0, \quad (22)$$

che possiamo riformulare come:

$$\lambda(1 - \lambda)\|\mathbf{x}_1 - \mathbf{x}_2\|^2 \geq 0, \quad (23)$$

che è facilmente verificabile.

La combinazione di una funzione convessa g con una mappa affine $g(\mathbf{A}\mathbf{x} + \mathbf{b})$ è convessa:

$$\begin{aligned} g(\mathbf{A}(\lambda\mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2) + \mathbf{b}) &= \\ g(\lambda(\mathbf{A}\mathbf{x}_1 + \mathbf{b}) + (1 - \lambda)(\mathbf{A}\mathbf{x}_2 + \mathbf{b})) &\leq \\ \lambda(\mathbf{A}\mathbf{x}_1 + \mathbf{b}) + (1 - \lambda)(\mathbf{A}\mathbf{x}_2 + \mathbf{b}) . \end{aligned} \tag{24}$$

Da questo risulta che il LS è convesso (ma non necessariamente strettamente convesso), e tutti i minimi sono globali (informalmente: immaginate un paraboloida nello spazio o una valle circondata da montagne uniformi).

Analizzando le equazioni normali, ci si accorge che il LS è speciale nel senso che le equazioni nel punto di stazionarietà descrivono un sistema di equazioni lineari che possono essere risolte in forma chiusa:

$$\frac{2}{n} \mathbf{X}^T (\mathbf{X}\mathbf{w} - \mathbf{y}) = \mathbf{0}. \quad (25)$$

Supponendo che la matrice $\mathbf{X}\mathbf{X}^T$ possa essere invertita (rango pieno):

$$\mathbf{w}^* = \underset{(d,d)}{\left(\mathbf{X}^T \mathbf{X}\right)^{-1}} \underset{(d,n)(n)}{\mathbf{X}^T} \mathbf{y} \quad (26)$$

Generiamo dati casuali:

```
1 # Simuliamo dati generati a loro volta da  
2 # un modello lineare (con rumore).  
3 X = np.random.randn((10, 5))  
4 y = X @ np.random.randn((5, 1)) + np.random.randn((10, 1))*0.01
```

Modello lineare:

```
1 w = np.random.randn((5, 1))  
2 yhat = X @ w # (10, 1)
```

Funzione costo:

```
1 mse = ((y - yhat)**2).mean()
```

Soluzione esplicita (numericamente instabile):

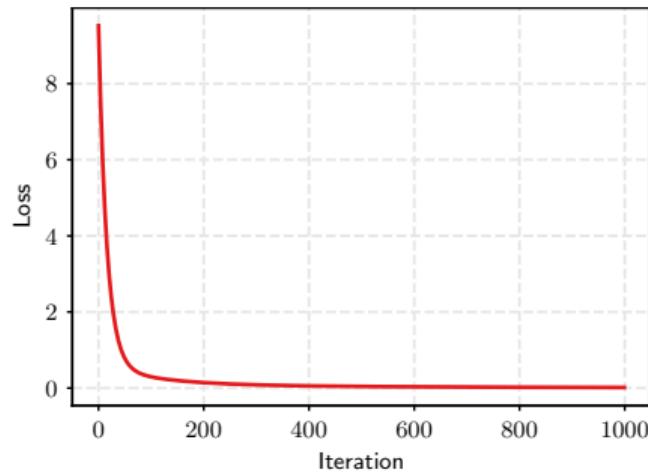
```
1 wopt = np.linalg.inv(X.T @ X) @ X.T @ y
```

Soluzione esplicita (numericamente stabile)

```
1 wopt = np.linalg.solve(X.T @ X, X.T @ y)
```

Implementazione base della discesa al gradiente:

```
1  for i in range(15000):
2      # Ricordando il segno meno nel gradiente...
3      w = w + 0.001 * X.T @ (y - X @ w)
```



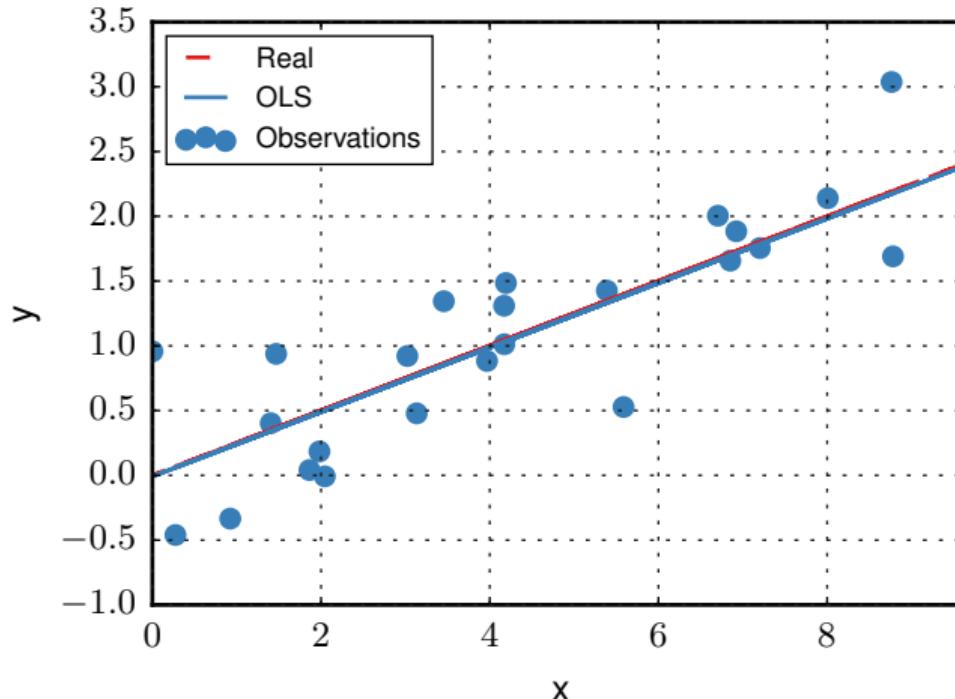


Figure 5: Si noti che in questo caso il LS è la soluzione *ottimale* ed *unbiased*.

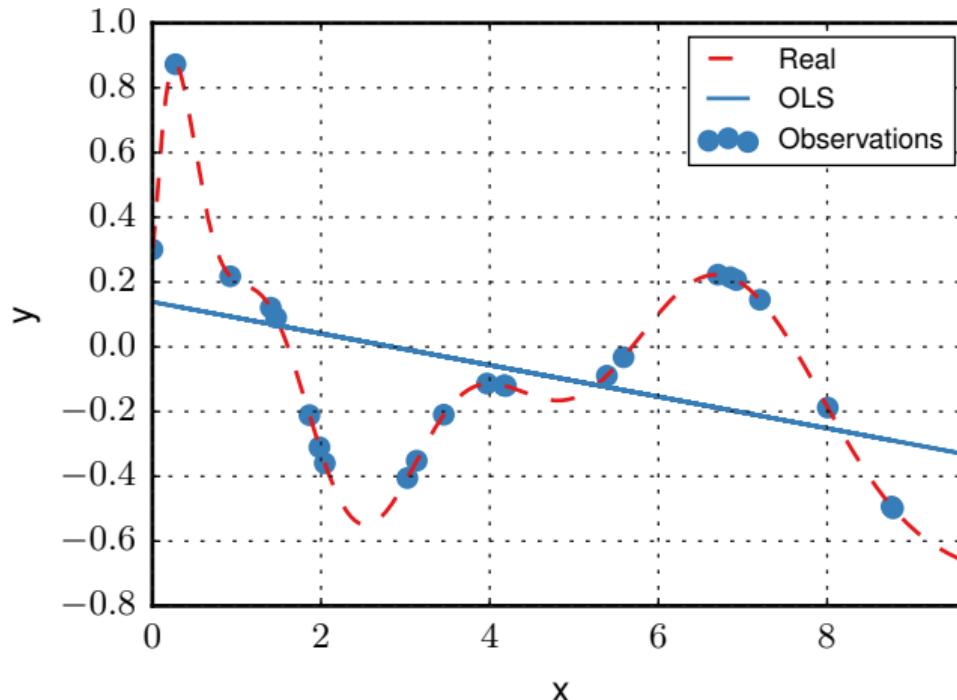


Figure 6: In un caso più complesso, le predizioni di un modello lineare sono molto limitate.

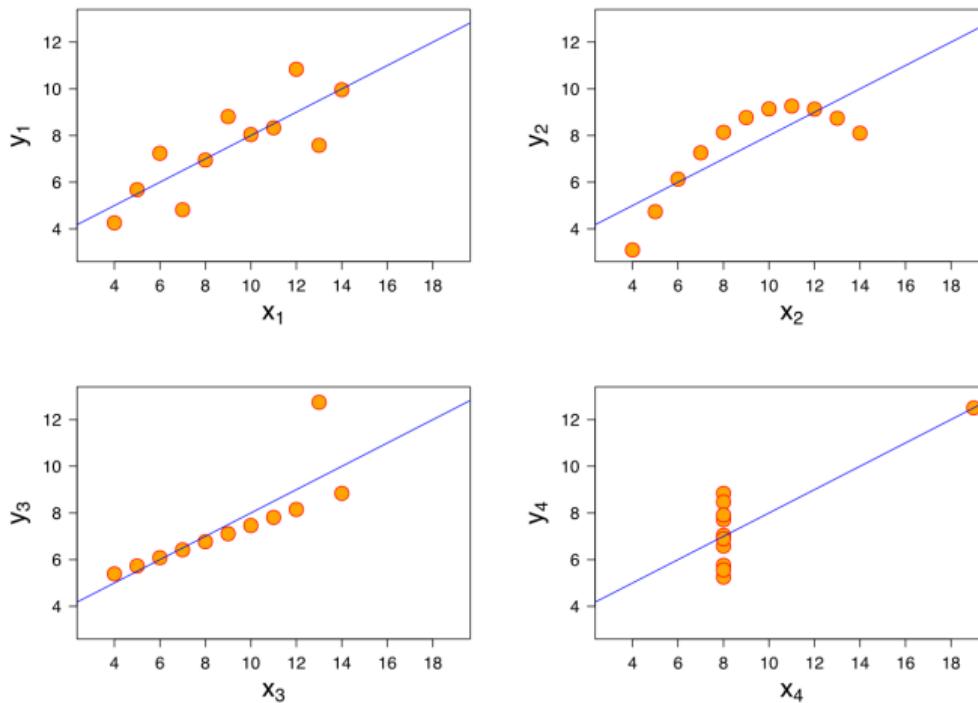


Figure 7: I quattro dataset del **quartetto di Ascombe** hanno la stessa soluzione.
Source: Wikimedia, author Schutz.

Oltre all'accuratezza predittiva, un aspetto fondamentale nella scelta di un algoritmo è il suo **costo computazionale**:

1. **Tempo**: quanto impiega l'algoritmo a trovare la soluzione?
2. **Memoria**: quanta RAM è necessaria per memorizzare i dati intermedi?

In particolare, ci interessa la **complessità asintotica**: non il tempo esatto in secondi (che dipende dall'hardware), ma come scala il costo al crescere della grandezza del dataset n e delle feature d .

Analizziamo il costo computazionale usando la notazione $\mathcal{O}(\cdot)$, che descrive come cresce il tempo di calcolo al crescere delle dimensioni del problema (n e d).

- ▶ **Soluzione esplicita:** Richiede calcolare $\mathbf{X}^\top \mathbf{X}$ in $\mathcal{O}(nd^2)$ e invertirla in $\mathcal{O}(d^3)$. Costo totale: $\mathcal{O}(nd^2 + d^3)$.
- ▶ **Discesa al gradiente:** Ogni passo richiede il calcolo del gradiente in $\mathcal{O}(nd)$. Per k passi: $\mathcal{O}(knd)$.

Se d (numero di feature) è molto grande, la soluzione esplicita è impraticabile (d^3), mentre la discesa al gradiente scala linearmente con d .