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, $\{1, \ldots, M\}$. 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:

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

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

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

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

In generale, non possiamo assumere che y_i sia determinato *univocamente* da $\mathbf{x}_i$ per diverse ragioni:

- Potrebbero esistere altre feature che determinano y_i che non osserviamo (**latent variables**).
- Alcune feature di $\boldsymbol{x}_i$ 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*:

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

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.

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

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 $p(\mathbf{x} \mid y)$ o $p(\mathbf{x})$ sono detti **generativi** (es., **linear discriminant analysis**).
2. Algoritmi che approssimano solo $p(y \mid \mathbf{x})$ 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, Stastistical learning theory, 1998

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

$$y^* = \arg\max_y p(y \mid \mathbf{x}) p(\mathbf{x}) = \tag{3}$$

$$\arg\max_y \log(p(y \mid \mathbf{x})) + \log(p(\mathbf{x})) = \arg\max_y \log(p(y \mid \mathbf{x}))\,. \tag{4}$$

Quindi, la cosa che ci interessa maggiormente è un modo di stimare il termine che massimizza la log-probability della classe.

Model-based supervised learning

Modelli lineari

Per proseguire, dobbiamo scegliere un modo di approssimare $p(y \mid \mathbf{x})$. In un approccio **model-based**, supponiamo che p sia definita da un insieme finito di valori predetti da una **funzione** $f(\mathbf{x})$.

Consideriamo il caso di regressione, e scegliamo la soluzione più semplice possibile: una funzione scalare $f(\mathbf{x})$ che predice la media di una Gaussiana, la cui varianza è fissata a σ^2:

$$p(y \mid \mathbf{x}) = \mathcal{N}(y \mid f(\mathbf{x}), \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y - f(\mathbf{x})}{\sigma}\right)^2\right) \tag{5}$$

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:

$$f(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + b = \mathbf{w}^\top \mathbf{x} + b\,, \tag{6}$$

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

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

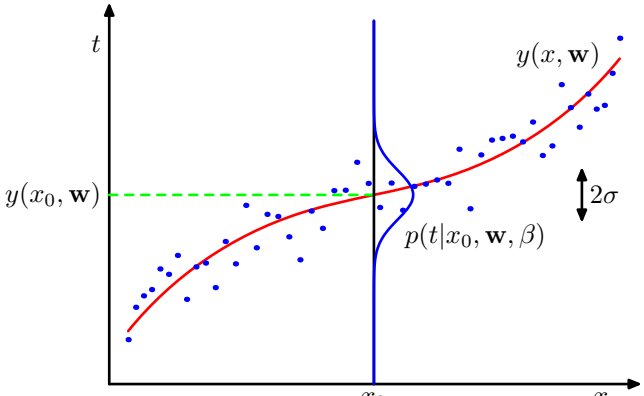


Figure 1: Riprodotto da Bishop e Bishop, 2023.

Supponendo che una feature del nostro input sia sempre costante, $\mathbf{x} = [x_1, \ldots, x_{d-1}, 1]$, 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 $\mathbf{w}$ del modello lineare? Intuitivamente, possiamo considerare i parametri che massimizzano la probabilità dei dati che abbiamo osservato (il nostro training dataset).

Questo criterio non è altro che il **maximum likelihood** applicato a questo scenario:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \prod_{i=1}^{n} p(y_i \mid f(\mathbf{x}_i), \sigma^2) = \arg\max_{\mathbf{w}} \sum_{i=1}^{n} \log\left(p(y_i \mid f(\mathbf{x}_i), \sigma^2)\right) \tag{7}$$

Una volta scelti p ed f, il problema può essere risolto tramite discesa al gradiente.

Mettendo tutto insieme, consideriamo quindi di ottenere la soluzione di maximum likelihood nel caso di regressione con una Gaussiana ed un modello lineare. Per cominciare, ricordiamo che:

$$\log(p(y \mid \mathbf{w}^\top \mathbf{x}, \sigma^2)) = -\frac{1}{2}\log(\sigma) - \frac{1}{2}\log(2\pi) - \frac{1}{\sigma^2}(y - \mathbf{w}^\top \mathbf{x})^2 \tag{8}$$

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

Sostituendo infine nel problema di partenza:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{\sigma^2} \sum_{i=1}^{n} \left(y_i - f(\mathbf{x}_i)\right)^2 \tag{9}$$

Il valore ottimale dei parametri $\mathbf{w}$ è quello che minimizza la somma dei residui al quadrato: per questa ragione, questo metodo viene detto **least-squares** regression.

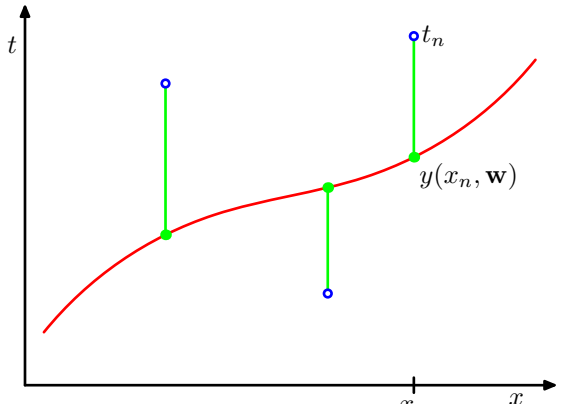


Figure 2: Scegliamo i parametri $\mathbf{w}$ che minimizzano la somma degli scostamenti (in verde) al quadrato.

Model-based supervised learning

Empirical risk minimization

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 $\hat{y}_2$.

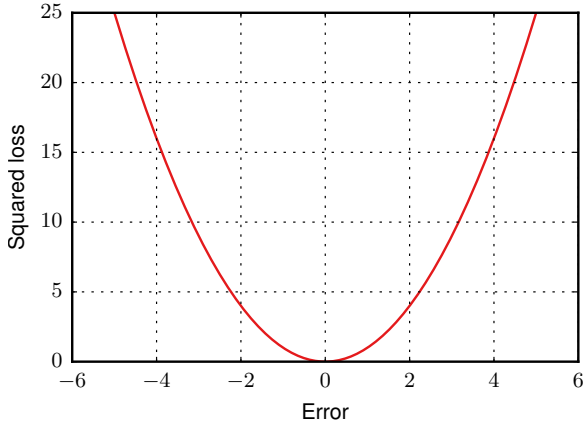


Figure 3: Visualizzazione della squared loss al variare del residuo $e = y - f(\mathbf{x})$.

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

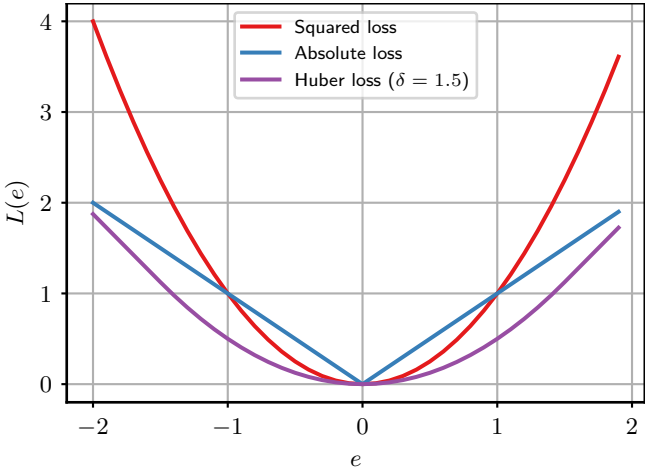
Ad esempio, per evitare di pesare quadraticamente errori sempre più alti, potremmo considerare il valore assoluto:

$$\text{Absolute deviation: } L(y, \hat{y}) = |y - \hat{y}| \,. \tag{10}$$

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

$$\text{Huber loss: } L(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{se } |y - \hat{y}| \leq \sigma \\ \sigma\left(|y - \hat{y}| - \frac{1}{2}\sigma\right) & \text{altrimenti} \end{cases} \tag{11}$$

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:

$$f^*(\mathbf{x}) = \arg\min_f \left\{ \mathbb{E}_{p(\mathbf{x},y)} \left[L(y, f(\mathbf{x})) \right] \right\} . \tag{12}$$

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**):

$$f^*(x) = \arg\min_f \left\{ \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) \right\} . \tag{13}$$

Abbiamo seguito due strade diverse per arrivare, sostanzialmente, alla stessa idea. In effetti, scelta p o scelta L, è possibile re-intepretarle come:

$$L(y, f(\mathbf{x})) = -\log p(y \mid f(\mathbf{x}))$$

I due approcci danno due prospettive diverse ed ugualmente interessanti: ad esempio, ragionare sulla differenza tra rischio atteso e rischio empirico è utile per studiare il comportamento di f, mentre ragionare su p permette di integrare misurare di incertezza nella predizione.

Per rendere più chiaro il confronto, ricordiamo che:

$$\arg\min_{\mathbf{w}} \; L(\mathbf{w}) = \arg\min_{\mathbf{w}} \; aL(\mathbf{w}) + b \tag{14}$$

per qualsiasi scelta di a e b. Quindi:

$$\arg\min_{\mathbf{w}} \frac{1}{\sigma^2} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 \tag{15}$$

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:

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

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

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

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

$$\nabla \mathrm{LS}(\mathbf{w}) = \frac{2}{n}\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y})\,. \tag{16}$$

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

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t \nabla \mathrm{LS}(\mathbf{w}_{t-1})\,. \tag{17}$$

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:

$$\|\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2\|^2 \leq \lambda\|\mathbf{x}_1\|^2 + (1-\lambda)\|\mathbf{x}_2\|^2 \tag{18}$$

Espandendo i termini:

$$-\lambda(1-\lambda)\mathbf{x}_1^\top \mathbf{x}_1 - \lambda(1-\lambda)\mathbf{x}_2^\top \mathbf{x}_2 + 2\lambda(1-\lambda)\mathbf{x}_1^\top \mathbf{x}_2 \leq 0\,, \tag{19}$$

che possiamo riformulare come:

$$\lambda(1-\lambda)\|\mathbf{x}_1 - \mathbf{x}_2\|^2 \geq 0\,, \tag{20}$$

che è facilmente verificabile.

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

$$\begin{aligned} g(\mathbf{A}(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) + \mathbf{b}) = \\ g(\lambda(\mathbf{A}\mathbf{x}_1 + \mathbf{b}) + (1-\lambda)(\mathbf{A}\mathbf{x}_2 + \mathbf{b})) \leq \\ \lambda(\mathbf{A}\mathbf{x}_1 + \mathbf{b}) + (1-\lambda)(\mathbf{A}\mathbf{x}_2 + \mathbf{b})\,. \end{aligned} \tag{21}$$

Da questo risulta che il LS è convesso (ma non necessariamente strettamente convesso), e tutti i minimi sono globali (informalmente: immaginate un paraboloide 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:

$$\frac{2}{n}\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y}) = \mathbf{0}\,. \tag{22}$$

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

$$\mathbf{w}^* = \underset{(d,d)}{\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}}\underset{(d,n)(n)}{\mathbf{X}^\top\,\mathbf{y}} \tag{23}$$

Generiamo dati casuali:

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

Modello lineare:

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

Funzione costo:

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

Soluzione esplicita (numericamente instabile):

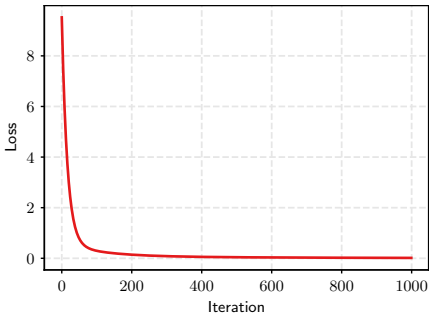
```
wopt = np.linalg.inv(X.T @ X) @ X.T @ y
```

Soluzione esplicita (numericamente stabile)

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

Implementazione base della discesa al gradiente:

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



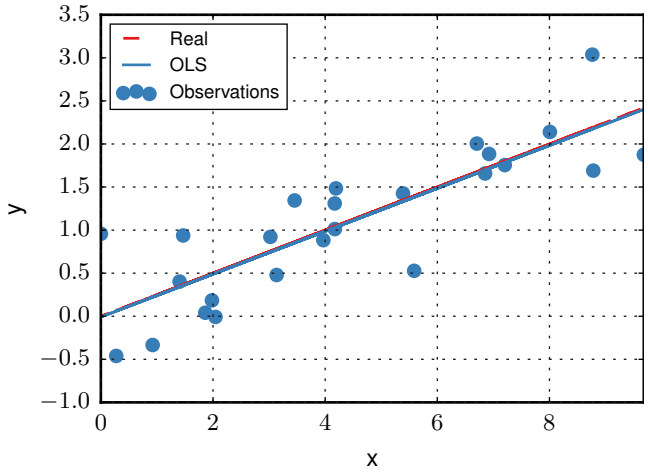


Figure 4: Si noti che in questo caso il LS è la soluzione *ottimale* ed *unbiased*.

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

$$\sigma^{2,*} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{w}^{*,\top}\mathbf{x})^2 \,. \tag{24}$$

In questo caso, l'incertezza del modello è fissa per ogni $\mathbf{x}$, ma è facile generalizzare a situazioni in cui anche σ^2 è funzione di $\mathbf{x}$ (modelli *eteroscedastici*).

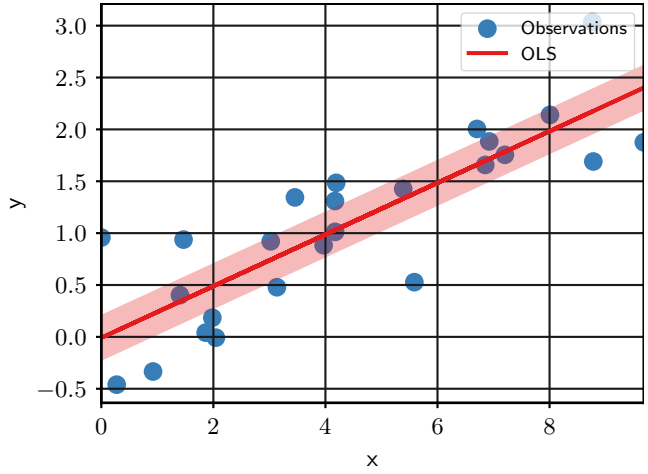


Figure 5: Stessa visualizzazione di prima, ma viene mostrata anche la banda corrispondente a più o meno σ.

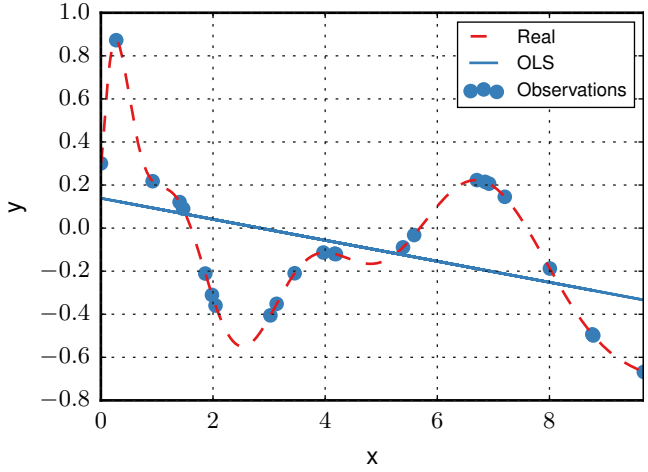


Figure 6: In un caso più complesso, le predizioni di un modello lineare sono molto limitate.

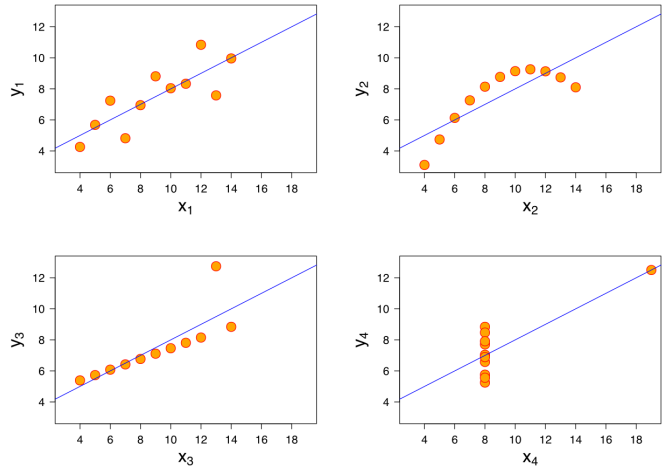


Figure 7: I quattro dataset del **quartetto di Ascombe** hanno la stessa soluzione.
Source: WikiMedia, author Schutz.

Ricordiamo che moltiplicare due matrici $\underset{(a,b)}{\mathbf{A}}\ \underset{(b,c)}{\mathbf{B}}$ ha una complessità $\mathcal{O}(abc)$, mentre l'inversione di una matrice ha complessità circa cubica..

Le operazioni da eseguire nella forma esplicita scalano cubicamente (inversione) o quadraticamente in d ed n, mentre le operazioni per la discesa al gradiente (se eseguite nell'ordine corretto) scalano solo *linearmente*.