Machine Learning

Emanuele Galiano
Damiano Trovato

Anno Accademico 2025/2026

Indice

Capitolo 1

Introduzione al Machine Learning

1.1 Definizione di Machine Learning (Arthur Samuel - 1959)

Nel 1959, Arthur Samuel fornisce una **definizione di machine learning**: il machine learning è il campo di studi che abilita i computer ad **imparare**, **senza essere esplicitamente programmati**.

Il paradigma alla base è fortemente diverso da quello tipico: se un algoritmo classico è **programmato ad hoc** per risolvere un task, e si comporta in maniera prettamente **deterministica**, un algoritmo di machine learning può **imparare a risolvere problemi** associati a vasti set di dati (classificare elementi, riconoscerli, trovare correlazioni). Questo permette di spostare il nostro focus non più sullo sviluppo di tutti gli step necessari affinchè un algoritmo risolvi **un problema specifico**, ma sulla creazione di un modello che riesca ad apprendere come risolvere **un insieme di problemi simili**.

1.2 Perchè abbiamo bisogno del Machine Learning

L'approccio utilizzato finora per risolvere i problemi è quello di:

- Trovare una logica per risolvere il problema.
- Scrivere un programma.
- Suddividerlo in pezzi più piccoli (funzioni).
- Automatizzare l'approccio.

Questo funziona per problemi di natura fortemente univoca, che sappiamo come risolvere, ad esempio:

- Computare l'area di un poligono.
- Risolvere equazioni differenziali.

Nel caso del poligono, supponendo di voler calcolare l'area di un rombo i dati presi in input sarebbero dati dalla coppia (x_1, x_2), contenente le lunghezze della diagonale principale e secondaria. Questi dati,passati ad un algoritmo, permettono di calcolarne l'area $\frac{x1*x2}{2}$ e generarne un output

Dati → Programma che risolve un task → Output

Alcuni problemi tuttavia presentano un alto grado di **incertezza**, che li rende più difficili da affrontare. Non poter fare assunzioni sui dati in input, e non conoscere tutti i possibili task, rende impossibile l'utilizzo di algoritmi standard per compiti del tipo:

- Classificazione di email spam e non spam
- Object detection

Il machine learning rappresenta la soluzione ideale a problemi di questo tipo, proponendo una nuova pipeline:

Dati + Output Atteso → Machine Learning → Soluzioni su nuovi dati

1.3 Esempio delle email spam e non-spam

Vogliamo creare un algoritmo di machine learning in grado di determinare se una mail è spam o meno. Il nostro obiettivo è quindi classificare ciascuna di queste come **spam**, o **ham**[1].

- `Compra prodotto a 10$! Oferta imperdibile!` → Spam
- `Ciao Giovanni, come stai?` → Ham

1.4 Machine Learning Algorithm (Tom Mitchel - 1998)

Un algoritmo **apprende dall'esperienza** E rispetto a una certa classe di **Task** T e a una misura di **performance** P. Se la sua **performance** nel compito T, misurata tramite P, migliora con l'**esperienza** E, allora quel modello ha appreso con successo.

1.5 Definizione di Task

Rappresenta il problema che deve essere risolto. Nell'esempio di determinare se una mail è spam o meno, il task è quello di **predire** l'etichetta (Y ="spam" oppure Y ="ham"), ed è strettamente legata al modello, che rappresentiamo come funzione parametrizzata, indicata con h_θ.

1.6 Definizione di Esperienza

Rappresentano i dati, ovvero i valori assunti dalle **random variables**, nell'esempio X è il contenuto della mail ed Y l'etichetta. La coppia di valori:

$$\{(X = x_i, Y = y_i)\}_{i=1}^{N}$$

[1] Email legittima, non spam.

Rappresenta l'esperienza. Generalmente vista come una collezione di elementi chiamati **esempi**.

1.7 Definizione di Performance

Funzione P che **valuta quanto bene** il modello è in grado di **risolvere un certo task** T. Supponiamo che il nostro algoritmo abbia previsto un insieme di etichette per un dato numero di email che indichiamo con:

$$\{\hat{y}_i\}$$

Dove il simbolo 'hat' indica che il dato non è stato osservato ma **previsto**. L'insieme delle etichette corrette è invece dato da

$$\{y_i\}$$

Per valutare la qualità del nostro metodo, dovremmo confrontare i due insiemi di previsioni utilizzando una **misura di performance:**

$$P(\{y_i\}, \{\hat{y}_i\})$$

Questa funzione restituisce un valore reale appartenente al range [0,1].

- Un **valore elevato** indica che le previsioni sono accurate
- Un **valore basso** indica che le previsioni non sono accurate.

Indichiamo con il termine **misura di errore** il valore: $1 - P$. Per risolvere problemi di machine learning ci affidiamo a modelli statistici che dipendono dal task.

1.8 Esempio completo

Siano:

- $x^{(1)}$: Il testo dell'email 1: "Compra prodotto a 10$! Oferta imperdibile!"
- $x^{(2)}$: Il testo dell'email 2: "Ciao Giovanni, come stai?"
- $y^{(1)}$: L'etichetta **spam**
- $y^{(2)}$: L'etichetta **ham**
- h: Il modello

Allora

$$h(x^{(1)}) = \hat{y}^{(1)}$$

e

$$h(x^{(2)}) = \hat{y}^{(2)}$$

1.9 Task, Esempi ed Etichette

Un esempio è generalmente espresso come una raccolta di valori che sono stati misurati quantitativamente da un evento osservato. Un esempio è generato da un vettore:

$$x \in \mathbb{R}^d$$

Scritto anche come:

$$x = (x_1, x_2, ..., x_d)$$

I valori del vettore x sono detti **features**, in quanto rappresentano **proprietà specifiche** degli esempi in input. Se la dimensionalità di x è 10, diremo che ha 10 features Nella maggior parte dei casi, ogni esempio x è anche abbinato a un output desiderato y. Tali output desiderati, sono anche chiamati **etichette**. Un'attività può quindi essere definita come un certo modo di elaborare un esempio di input per ottenere un output.

Torniamo al nostro esempio: determinare se un'e-mail è spam o ham. In questo caso, l'input è l'email, le features possono essere caratteristiche dell'email, come il numero di errori ortografici o la presenza di alcune parole chiave, mentre l'output atteso è l'etichetta (spam o ham).

1.10 Estrazione delle features

Per gestire le email, dobbiamo prima trasformarle in un'**entità quantificabile**. Questo di solito viene fatto **identificando alcune caratteristiche** dei dati che sono **rilevanti per il compito dato** (numero di errori ortografici o la presenza di alcune parole chiave). In pratica, stiamo cercando una funzione f che trasformi l'entità dalla sua forma originale a una forma di destinazione, che è buona per risolvere un compito specifico:

$$x \rightsquigarrow f(x) \rightsquigarrow \overline{x}$$

Dove x è il dato grezzo di input (ad esempio, il messaggio di posta elettronica completo), f è la funzione di trasformazione e $\overline{x}$[2] è l'output della trasformazione, che sarà l'input dell'algoritmo di apprendimento automatico.

La funzione f è chiamata *rappresentazione*. L'output della trasformazione x è anche chiamato rappresentazione. Poiché rappresentando i dati otteniamo un vettore di funzionalità, il processo di rappresentazione dei dati è talvolta chiamato **features extraction**. Non ci sono «rappresentazioni universali», ma solo rappresentazioni che servono a qualche compito.
Le rappresentazioni sono di 2 tipi:

- Create a mano
- Apprese

[2]Su questa notazione: da ora in poi, quando ci riferiremo all'output della trasformazione, non useremo più $\overline{x}$, ma direttamente x, dando per scontato il passaggio di rappresentazione $f(x)$.

L'estrazione delle features **mette in luce caratteristiche salienti** trascurandone altre.

1.11 Features

Generalmente, l'output di una funzione di rappresentazione è nella forma:

$$x = (x_1, x_2, \ldots, x_d), \quad x \in \mathbb{R}^d$$

Questo, è composto da un insieme di features . **Una feature è la specifica di un attributo**. Si tratta di una misura che rappresenta **aspetti dei dati** che è utile **evidenziare per risolvere il problema considerato**. Ad esempio, il colore può essere un attributo. "Il colore è blu" è una funzionalità estratta da un esempio.
Le caratteristiche possono essere di due tipi principali:

Categoriche: un numero finito di valori discreti. Questi possono essere:

- **Nominali:** a indicare che non esiste **alcun ordinamento** tra i valori, ad esempio cognomi e colori.
- **Ordinali:** a indicare che esiste un **ordinamento rilevante**, ad esempio in un attributo che assume i valori basso, medio o alto.

Continue: comunemente, **sottoinsieme di numeri reali**, dove c'è una differenza misurabile tra i valori possibili. I numeri interi sono solitamente trattati come continui nei problemi pratici.

1.12 Esempio delle Email Spam e Non Spam

Consideriamo il nostro esempio in cui vogliamo distinguere le e-mail spam da quelle non spam. L'input del processo sono i messaggi di posta elettronica, quindi dobbiamo trasformarli in vettori di features:

$$x = (x_1, x_2, ..., x_n)$$

con un processo di *features extraction.*

Naturalmente, ci aspettiamo che le funzionalità estratte siano utili per risolvere il nostro compito di determinare se un'e-mail è spam o ham. Possiamo notare che le e-mail di spam spesso includono errori ortografici e parole come "Acquista", "occasione" e "10$". Quindi, potremmo decidere di rappresentare ogni messaggio di posta elettronica con due numeri:

- Il conteggio degli errori ortografici.
- Il numero di volte in cui alcune parole o pattern specifici appaiono nel testo.

Una volta che i messaggi di input sono stati convertiti in vettori di funzionalità, possono essere visti come vettori nello spazio 2D.

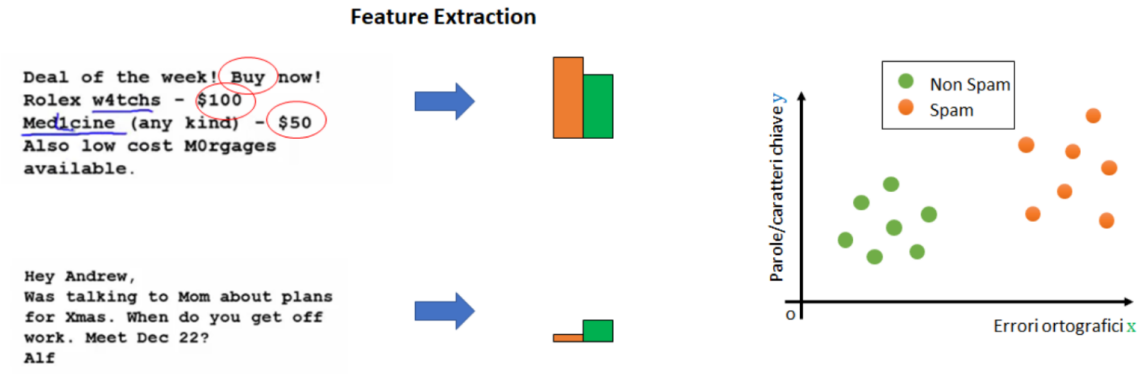

Figura 1.1: Estrazione delle feature: le e-mail vengono trasformate in vettori 2D, dove x = errori ortografici e y = pattern ripetibili, e proiettate nello spazio delle feature, , dove la separazione tra spam (arancione) e non spam (verde) risulta evidente.

1.13 Tipologie di Task

Le attività possono essere di diversi tipi. Di seguito, discuteremo due compiti principali:

- Classificazione
- Regressione

Assumeremo che ogni algoritmo di apprendimento automatico prenda come input esempi che sono già stati rappresentati con una funzione di rappresentazione adeguata.

1.14 Classificazione

In questo tipo di attività, alla macchina viene chiesto di specificare a quale di un insieme predefinito di categorie K appartiene l'input.
Esempi di questo compito sono:

- Classificare i post di Facebook come riguardanti la politica o qualcos'altro (classificazione politica vs non politica).
- Rilevamento delle e-mail di spam (classificazione dello spam vs legittima delle e-mail).
- Riconoscimento dell'oggetto raffigurato in un'immagine tra 1000 oggetti diversi (riconoscimento dell'oggetto).

L'algoritmo di apprendimento è solitamente fornito con un insieme di esempi:

$$\{x^{(1)}, x^{(2)}, ..., x^{(n)}\} \text{ dove: } x^{(j)} \in \mathbb{R}^N \forall j$$

e un insieme di etichette corrispondenti

$$\{y^{(1)}, y^{(2)}, ..., y^{(n)}\} \text{ dove: } y^{(j)} \in \{1, .., k\} \forall j$$

che specificano a quale delle categorie K appartiene ogni esempio.

Ad esempio, se $y^{(j)} = 3$, allora $x^{(j)}$ appartiene alla classe "3".

Nel caso della classificazione binaria (ad esempio, spam vs non spam), $y^{(j)} \in \{0, 1\}$. Per risolvere questo compito, l'algoritmo di apprendimento automatico assume la forma di una funzione:

$$h : \mathbb{R}^N \to \{1, ..., K\}$$

tale che:

$$y^{(j)} = h(x^{(j)})$$

Esempio:

- **Classification Task:** data un'e-mail, classificarla come spam o non spam.
- **Input:** esempi n-dimensionali $x = (x_1, x_2, ..., x_n)$ contenenti le caratteristiche dell'email, come il numero di errori ortografici e l'occorrenza di parole specifiche.
- **Output:** etichette $y \in \{0, 1\}$ che indicano se l'e-mail è legittima o spam.

1.15 Regressione

In questo tipo di compito, al programma del computer viene chiesto di prevedere un valore numerico dato un input, tipo:

- Prevedere il prezzo delle case date alcune caratteristiche come la città, l'età, la zona, ecc.
- Prevedere il valore futuro delle azioni di una società dai valori di altre società o da altre statistiche sul mercato (previsione del mercato azionario).
- Conta il numero di auto presenti in un'immagine.

Analogamente alla classificazione, l'algoritmo viene fornito con esempi di training $x \in \mathbb{R}^N$ e con gli output desiderati $y \in \mathbb{R}$. L'algoritmo di apprendimento automatico assume la forma di una funzione $h : \mathbb{R}^N \to \mathbb{R}$ tale che $y^{(j)} = h(x^{(j)})$.
Esempio:

- **Regression task:** Predire il prezzo di una casa in base ai suoi metri quadrati.
- **Input:** Dimensione della casa x (valore scalare)
- **Output:** Prezzo y.

1.16 Supervised Learning e Unsupervised Learning

Gli approcci di Machine Learning possono essere approssimativamente divisi in supervised e unsupervised learning.

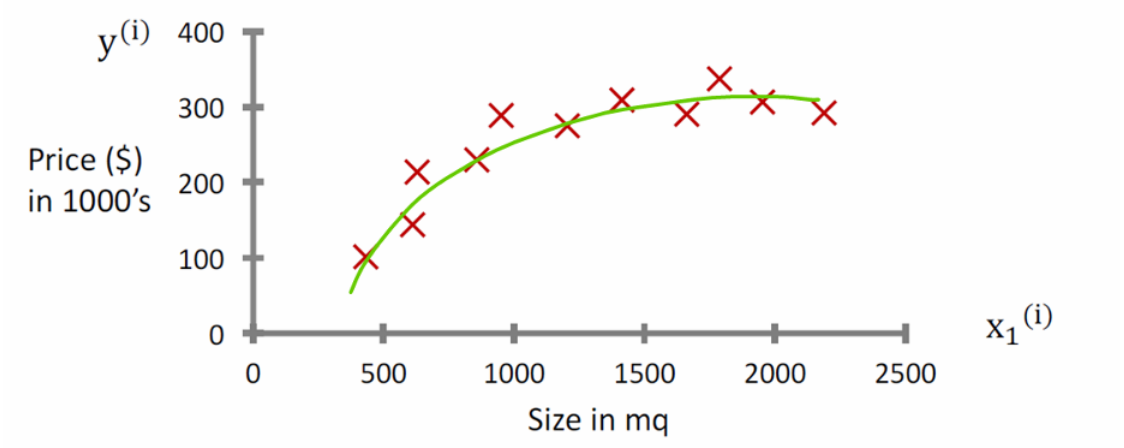

Figura 1.2: Relazione tra dimensione dell'immobile ($x_1 1^{(i)}$, in mq) e prezzo ($y^{(i)}$, in migliaia di \$): i punti rossi sono i dati osservati, la linea blu rappresenta un modello di regressione lineare che non approssima bene l'andamento non lineare.

Supervised Learning: L'algoritmo viene addestrato su un insieme di esempi di input e output desiderati. L'obiettivo è imparare una funzione che mappi gli input agli output corretti.

Unsupervised Learning: L'algoritmo viene addestrato solo su esempi di input, senza output desiderati. L'obiettivo è scoprire la struttura o i pattern nei dati.

$$\{x^{(1)}, x^{(2)}, ..., x^{(n)}\} \text{ dove: } x^{(j)} \in \mathbb{R}^N \forall j$$

Questi tipi di compiti mirano generalmente a modellare la struttura dei dati. Un esempio di unsupervised learning è il clustering, in cui non viene fornita alcuna informazione aggiuntiva oltre agli esempi.

Gli approcci supervised sono generalmente più facili da gestire, ma richiedono la presenza di labels. Ottenere labels è spesso un problema costoso in termini di tempo, poiché richiede che le persone annotino manualmente i dati. Ad esempio, se dobbiamo costruire un spam-detector utilizzando un approccio supervised, è necessario che qualcuno etichetti manualmente diverse email come 'spam' o 'non-spam'.

1.17 Reinforcement Learning

Alcuni autori fanno riferimento anche a una terza classe di algoritmi di Machine Learning: il Reinforcement Learning.

Il Reinforcement Learning mira a scoprire la soluzione a un problema attraverso il metodo trial and error, piuttosto che tramite istruzioni esplicite su come risolvere il compito. Questo avviene permettendo all'algoritmo di interagire con un environment e ricevere positive rewards quando compie azioni che portano a un buon risultato (rispetto al problema da risolvere) e negative rewards quando compie azioni che portano a un risultato negativo.

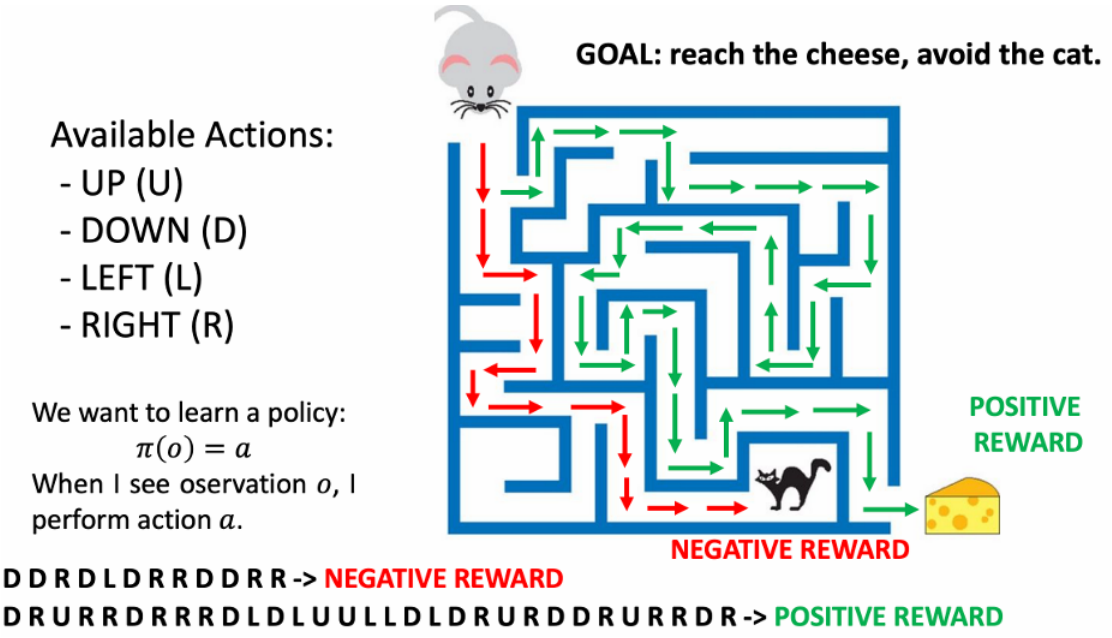

Figura 1.3: Reinforcement Learning nel labirinto: l'agente sceglie tra azioni U/D/L/R e apprende una policy $\pi(o) = a$ che massimizza la ricompensa, raggiungendo il formaggio (positiva) ed evitando il gatto (negativa).

L'obiettivo degli algoritmi di Reinforcement Learning è apprendere una policy π, che possa essere utilizzata per determinare quale azione a intraprendere quando si acquisisce un'osservazione del mondo o. Questo processo ricorda il modo naturale in cui gli animali imparano a risolvere problemi. Ad esempio, si può pensare a un topo che deve trovare l'uscita da un labirinto (immagine 1.3).

1.18 Misura di Performance (P)

Per valutare le capacità di un algoritmo di Machine Learning nel risolvere un determinato compito, è necessaria una misura quantitativa delle sue prestazioni. Solitamente, questa performance measure P è specifica per il task T che il sistema sta eseguendo.

Per compiti come la classification, spesso si misura la performance utilizzando l'**accuracy**, ovvero la percentuale di esempi classificati correttamente dal modello. Nel caso della regression, invece, si possono usare altre metriche come il mean squared error.
Le misure di performance sono utilizzate per due motivi principali:

- Capire quando un algoritmo di Machine Learning sta migliorando in un determinato compito.
- Valutare la performance dell'algoritmo una volta finalizzato.

Una performance measure può anche essere vista in termini di error. Ad esempio, l'**accuracy** corrisponde a un error rate (la percentuale di esempi classificati in modo errato), calcolato come $1 - accuracy$.

1.18.1 Esempio

Un spam detector analizza cinque email. Le prime tre sono spam, le ultime due non lo sono. L'algoritmo classifica come spam le prime due email e come non spam le ultime tre. In questo caso, la prima e le ultime due classificazioni sono corrette, mentre la terza è errata. La accuracy si calcola come la percentuale di esempi classificati correttamente:

$$\frac{4}{5} = 0.8 \quad \text{ovvero} \quad 80\%$$

1.19 Experience (E)

Un algoritmo di Machine Learning apprende dall'**experience** per migliorare una performance measure su un determinato task.
L'**experience** è costituita da una raccolta di esempi

$$\mathbf{x}^{(i)}$$

(noti anche come data points, poiché possono essere mappati in uno spazio multi-dimensionale tramite una funzione di rappresentazione), eventualmente accompagnati dalle relative labels

$$y^{(i)}$$

(a seconda del task considerato).
Esistono due principali tipi di algoritmi di Machine Learning:

- Supervised approaches (quando abbiamo le paired labels, ad esempio nella classification e nella **regression**).
- Unsupervised approaches (quando non abbiamo paired labels, come nel **clustering**).

L'**experience** assume forme diverse a seconda del tipo di approccio di Machine Learning utilizzato.

1.20 Dataset (D)

Le performance measures vengono generalmente calcolate rispetto a un insieme di esempi, piuttosto che su singoli esempi. Un insieme di esempi (eventualmente con labels) è chiamato **dataset**. I datasets sono generalmente omogenei, nel senso che i dati contenuti al loro interno hanno un formato simile. Ad esempio:

- Nel Fisher's Iris dataset, tutti gli esempi hanno 4 features e una label corrispondente a una delle tre classi.
- In un dataset di immagini di food, ogni immagine è associata a una class che indica il piatto specifico.

1.21 Design Matrix

Un modo comune per rappresentare un dataset è utilizzare una design matrix. Poiché ogni esempio è una collezione di **n** features, un dataset di **m** elementi può essere rappresentato tramite una matrice

$$\mathbf{X} \in \mathbb{R}^{m \times n}$$

con dimensione m x n.

- Ogni riga della design matrix rappresenta un esempio.
- Ogni colonna rappresenta una delle features.

Nel caso del supervised learning, si considera spesso anche un'altra matrice

$$\mathbf{Y} \in \mathbb{A}^{m \times k}$$

dove k è la dimensionalità degli output desiderati.
Ad esempio, nel caso della classification,

$$\mathbb{A} = \{1, \ldots, M\}$$

dove M è il numero di classi e k è spesso uguale a 1.

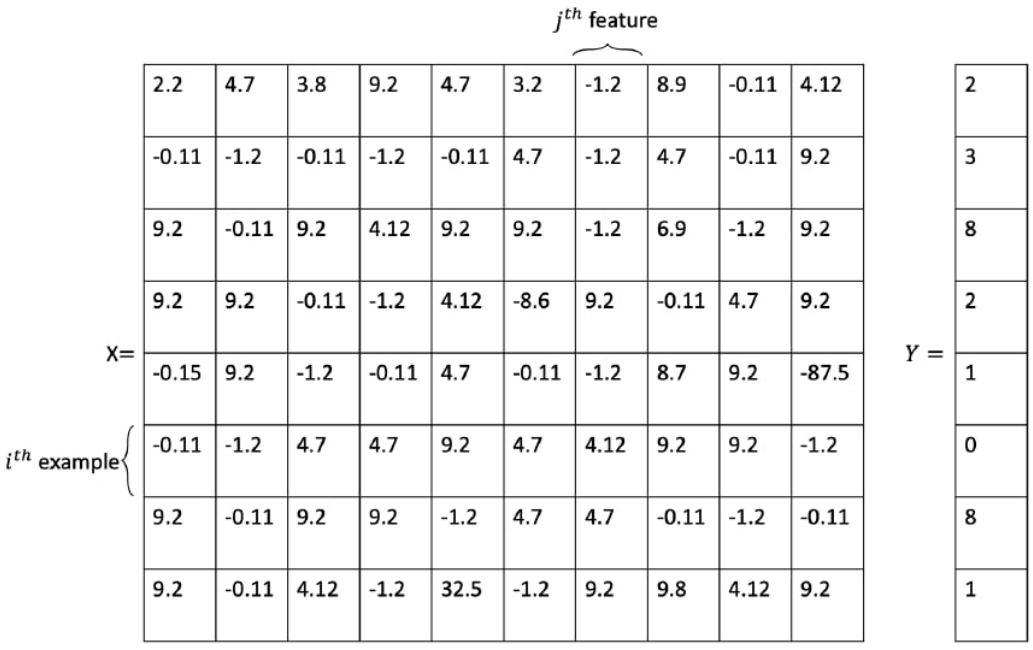

Figura 1.4: Design Matrix: rappresentazione di un dataset con m esempi e n features. Ogni riga corrisponde a un esempio, ogni colonna a una feature.

1.21.1 Esempio

Supponiamo di avere un dataset composto da 1000 email, alcune classificate come spam e altre come not spam. Assumiamo che ogni email sia rappresentata da due features, come discusso nei precedenti esempi.
La design matrix che rappresenta il dataset è una matrice

$$\mathbf{X} \in \mathbb{R}^{1000\times 2}$$

- Ogni elemento della matrice rappresenta una delle features di un esempio nel dataset.
- Ad esempio, $X_{i,1}$ indica il numero di errori ortografici nell'**i-esima email**, mentre $X_{j,2}$ rappresenta il numero di occorrenze di parole chiave nell'**j-esima email**, e così via.

Le labels sono contenute in un vettore

$$\mathbf{Y} \in \{0,1\}^{1000}$$

dove:

- Y_i rappresenta la label associata all'**i-esimo esempio** (ad esempio, 0 = not spam, **1 = spam**).

1.22 Learning

Un algoritmo di Machine Learning utilizza un dataset di esempi per migliorare la sua performance in un determinato task. Il processo di miglioramento della performance dell'algoritmo è chiamato learning o training.

1.22.1 In cosa consiste il training?

- Un algoritmo di Machine Learning ha alcuni parametri chiamati parameters, che possono essere regolati per modificarne il comportamento. Questi parametri sono legati a un model (una funzione matematica) utilizzata per risolvere il task.
- Un algoritmo chiamato training procedure utilizza gli esempi forniti per trovare i valori ottimali per questi parameters.
- Alcuni parametri non possono essere regolati automaticamente dal training. Questi sono detti hyperparameters e devono essere ottimizzati al di fuori della training procedure, spesso attraverso un metodo trial and error.

1.22.2 Esempio

Consideriamo un semplice spam detector che classifica le email come spam o non-spam in base al numero di errori ortografici.
L'algoritmo può essere scritto come segue:

```
def classify(x):
    if x > A:
        return 1  # Spam
    else:
        return 0  # Non-spam
```

L'algoritmo dipende da un singolo parametro A. La domanda è: quale valore dovremmo assegnare a A? La *training procedure* permette di trovare un valore adatto per A. Una semplice training procedure consisterebbe nel provare diversi valori per A e registrare le performance dell'algoritmo per ciascun valore di A. Alla fine, possiamo scegliere il valore di A che massimizza la performance measure P.