

# Machine Learning

Albert SAMUEL 1859 - Definizione

Machine Learning de ei comprender l'abilità di imparare senza essere programmate esplicitamente

Un algoritmo di apprendimento automatico può imparare a risolvere un problema specifico a partire da un insieme di dati.

FOCUS => Progettare tutti i passaggi necessari per risolvere un problema alla creazione di algoritmi che possono efficacemente imparare come risolverlo.  
NUOVO

Programmazione classica => Progettazione di una procedura da cui si può seguire per risolvere un esempio specifico

Apprendimento automatico => Imparato da una serie di tecniche per istruire un gruppo di passi apprendendo da sole come risolvere un bello problema.

Generalmente pensiamo a un approccio logico per risolvere un problema, scriviamo un programma, eventualmente suddiviso in diverse parti, che ci permetta di automatizzare l'approccio logico che abbiamo ideato.

Campione: area di un poligono

$$\text{Rombi } D_p \cdot D_s / 2 \quad \text{Dati} \rightarrow \text{Algoritmo} \rightarrow \text{Risultato}$$

Alcuni problemi tuttavia sono particolarmente difficili e implicano incertezza. Vi sono problemi molto difficili da modellare come classificare un'immagine per determinare quale oggetto vi è raffigurato oppure distinguere e-mail legittime da email di spam

Questa seconda classe di problemi è molto più difficile da gestire. Da un lato, ciò è dovuto al fatto che questi fenomeni sono governati da una grande incertezza, che non ci permette di fare molte assunzioni sui dati che probabilmente vedremo. Dall'altro lato, anche se, come esseri umani, possiamo risolvere queste attività, non sempre sappiamo come lo facciamo. Ad esempio, possiamo capire quali oggetti sono presenti in un'immagine, ma il modo in cui il nostro cervello lo fa è ancora poco conosciuto.

Per comprendere meglio tutte le definizioni fornite in questa lezione, considereremo un esempio semplice in cui vogliamo creare un algoritmo di apprendimento automatico per determinare se una data email è spam o meno. Diremo che ogni email può andare in due "contenitori" (che chiameremo in seguito "classi"): spam e ham

From: cheapsales@buystufffromme.com  
To: ang@cs.stanford.edu  
Subject: Buy now!

Deal of the week! Buy now!  
Rolex w4tchs - \$100  
Medicine (any kind) - \$50  
Also low cost M0rgages available.

Spam

From: Alfred Ng  
To: ang@cs.stanford.edu  
Subject: Christmas dates?

Hey Andrew,  
Was talking to Mom about plans  
for Xmas. When do you get off  
work. Meet Dec 22?  
Alf

Non-spam

E-mail 1 (Potenziale Spam): Simboli ed errori ortografici

E-mail 2(Non Spam): Più confidenziale e personale

Osservando queste email notiamo che la prima presenta simbolo strano simbolo del dollaro in un testo italiano e un'offerta sospetta. La seconda sembra una comunicazione normale. Tuttavia non è facile definire una regola logica generale basata solo su questo.

Estrazione delle Caratteristiche (Feature Extraction): Poiché il contenuto grezzo delle email può essere vario e contenere informazioni irrilevanti, è necessario un processo di estrazione delle caratteristiche (feature). Questo processo consiste nel trovare una rappresentazione dell'email sotto forma di un vettore di numeri, dove ogni numero (feature) cattura un aspetto saliente dell'email rilevante per la classificazione.

Esempi di Feature:

- Numero di errori ortografici: Le email di spam spesso contengono errori grammaticali o di battitura.
- Presenza di parole chiave specifiche: Parole come "offerta imperdibile", "guadagna facilmente", "clicca qui" o riferimenti a valute o sconti elevati potrebbero essere indicatori di spam. La parola "dollar" nell'esempio potrebbe essere una keyword sospetta in un testo italiano.
- Frequenza di determinati caratteri o simboli: L'uso eccessivo di punti esclamativi, lettere maiuscole o simboli come il dollaro in contesti inappropriati.

La funzione di rappresentazione ( $f$ ) prende l'email come input e produce un nuovo vettore ( $x$  soprasseguito) contenente queste caratteristiche. Ad esempio, un'email potrebbe essere rappresentata da un vettore con due componenti: il numero di errori ortografici e il numero di parole chiave sospette.

Spazio delle Caratteristiche: Una volta estratte le caratteristiche, ogni email può essere vista come un punto in uno spazio multidimensionale (dove ogni dimensione corrisponde a una feature). L'idea è che le email di spam e non spam occupino regioni diverse in questo spazio.

Il Modello (Classificatore): L'obiettivo è apprendere una funzione (modello)  $H$  che prenda in input il vettore delle caratteristiche di una nuova email e produca in output la sua classe (spam o non spam). Questo modello può essere concettualizzato come una linea (in uno spazio 2D) o una superficie (in spazi multidimensionali) che separa le email di spam da quelle non spam.

Apprendimento Supervisionato: Per addestrare il modello, si utilizza un approccio di apprendimento supervisionato. Ciò significa che l'algoritmo di apprendimento riceve un insieme di dati di training composto da email già etichettate come spam o non spam. L'algoritmo analizza questi dati per trovare i parametri ottimali del modello che gli permettano di generalizzare bene a nuove email.

Misura delle Performance (Accuratezza): Per valutare quanto bene il modello apprende e generalizza, si utilizzano delle misure di performance. Nel caso della classificazione (come spam/non spam), una metrica comune è l'accuratezza, che rappresenta la percentuale di email classificate correttamente. Nell'esempio fornito, su cinque email con etichette reali (spam, spam, spam, non spam, non spam) e predizioni del modello (spam, spam, spam, non spam), l'accuratezza è di  $4/5 = 0.8$  (80%) perché quattro email su cinque sono state classificate correttamente.

Generalizzazione e Valutazione: È fondamentale valutare le performance del modello su dati mai visti prima (un dataset di test) per verificare la sua capacità di generalizzare e non semplicemente memorizzare i dati di training. Se il modello ha un'alta accuratezza sui dati di training ma bassa sui dati di test, potrebbe esserci un problema di sovradattamento (overfitting). In alcuni casi, si utilizza anche un dataset di validazione per la selezione degli iperparametri del modello o per confrontare diversi modelli.

## Problema del Learning → Lettura esercizio del cassetto di apprendimento autonome

Torna parte della considerazione che la precedente definizione non è buona posta

Un computer apprende da un'esperienza  $E$  rispetto ad un task  $T$  su una matrice di personale D, se le parametri D misurate su  $E$  migliora con l'esperienza rispetto ad un task  $T$



**TASK** → Questo è il problema da risolvere. Nell'esempio delle e-mail spams o ham, predire l'etichetta ( $Y = \text{SPAM}$  ovvero  $Y = \text{HAM}$ ) dato che mail è il test.

**TASK RAPPRESENTA IL PROBLEMA SPECIFICO CHE SI DESIDERÀ RISOLVERE AD UN VERSO APPRENDIMENTO AUTONOMO.**

**STIMAMENTO** collegato al modello che è una funzione parametrizzata ( $h$ )

**ESPERIENZA** → Questo è il dataset. Doti → valori estratti da variabili discrete. Nel nostro esempio se  $X$  contenuto dell'email e  $Y$  le sue etichette, un insieme di coppie di valori  $\{(X = x_i, Y = y_i)\}_{i=1}^n$  costituisce l'esperienza. L'esperienza viene generalmente vista come due raccolte di elementi (coppie di valori in questo caso). Ogni elemento è anche chiamato "esempio".

Esperienze DEVONO ESSERE RILUVANTI PER LO SPECIFICO TASK che si vuole risolvere.

Nel nostro caso esperienza i contributi dei email (il corpo delle mail) e delle loro etichette associate (spam o non spam)

Nel contesto di un dataset, questi esempi possono essere organizzati in una matrice di design. In questa matrice, le righe corrispondono ai singoli esempi, mentre le colonne rappresentano gli attributi o le feature di ciascun esempio. Ogni elemento all'interno di questa matrice rappresenta il valore della  $j$ -esima caratteristica dell' $i$ -esimo esempio

## PBL. PERFORMANCE

→ Misura di prestazione: questa è una funzione che valuta quanto bene il computer riesce a risolvere la task  $T$ . Supponiamo che il nostro algoritmo abbia previsto un insieme di etichette per un certo numero di email:  $\{\hat{y}_i\}_i$  (qui il simbolo "cappello" indica che questo valore non è stato osservato, ma lo abbiamo preveduto) e supponiamo che le etichette corrette siano l'insieme  $\{y_i\}_i$ . Per valutare quanto sia buona la nostra metodologia, dovremmo confrontare i due insiemi di previsioni utilizzando una misura di prestazione  $P(\{y_i\}_i, \{\hat{y}_i\}_i)$ . Questa funzione generalmente restituisce un valore reale. Un valore alto (alta prestazione) indica che le previsioni sono accurate, mentre un valore basso (bassa prestazione) indica che le previsioni non sono accurate. Ad esempio,  $P$  potrebbe semplicemente calcolare il numero di email classificate correttamente. Possiamo definire diverse misure di prestazione. Vedremo le principali in futuro.

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

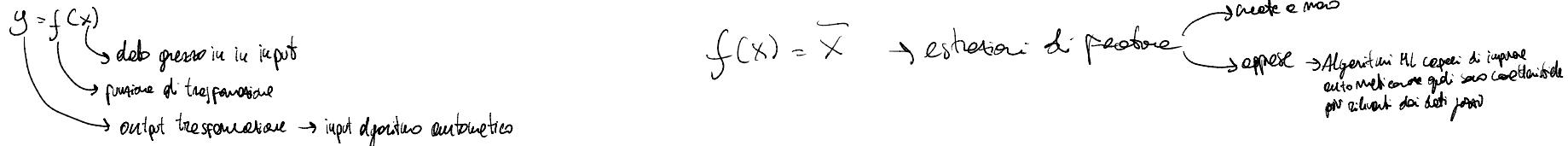
$$P \in [0, 1]$$

$1 - P \Rightarrow$  Minore di zero → buone previsioni!  
Acumulatore

Modello  $\rightarrow h_0$  pure utile

Funzione di valutazione modello se sono imposto dato in input val. 1  
 $P(\{y^{(i)}\}_{i=1}^m, \{\hat{y}^{(i)}\}_{i=1}^m)$  →  $0 \rightarrow S$  buone previsioni  
 $1 \rightarrow C$  cattive previsioni

Abbiamo visto che un algoritmo di apprendimento automatico generalmente si aspetta che gli esempi siano sotto forma di vettori di feature. Tuttavia nella maggior parte dei casi gli esempi arrivano in una forma che non è molto comoda da elaborare. Per lavorare con queste entità dobbiamo prima trasformarle in qualche entità quantificabile. Questo viene fatto identificando alcune caratteristiche dei dati che sono importanti per il compito dato. In pratica, stiamo cercando una funzione  $f$  che trasforma l'entità della sua forma originale a una forma finale, adatta per risolvere un compito specifico:



La funzione  $f$  è chiamata rappresentazione. Anche l'output della trasformazione  $f$  è chiamato rappresentazione. Quando  $y$  è multidimensionale (chiamato anche multivariato), viene spesso definito come un vettore di caratteristiche. Il fatto che sia  $f$  che  $y$  siano chiamati "rappresentazione" può generare confusione. Tuttavia, spesso è chiaro dal contesto a quale dei due concetti ci riferiamo. Poiché rappresentando i dati otteniamo un vettore di caratteristiche, il processo di rappresentazione dei dati è talvolta chiamato **estrazione delle feature** caratteristiche.

Le funzioni di rappresentazione sono fondamentali per lavorare con l'apprendimento automatico. Senza di esse, i dati sarebbero spesso in una forma semplicemente impossibile da elaborare. In generale, le rappresentazioni sono funzioni dei dati che rendono esplicite alcune cose sui dati, trascurando altri fattori, a seconda del compito considerato.

Poiché dobbiamo capire cosa trascurare e cosa rendere esplicito, quando creiamo una rappresentazione, dobbiamo tenere a mente un compito. Non esistono "rappresentazioni universali", solo rappresentazioni che sono buone per un certo compito.

## CARATTERISTICHE

L'output di una funzione di rappresentazione è quindi in generale un esempio  $\langle \#_1, \dots, \#_n \rangle$  composto da un insieme di caratteristiche  $x_i$ . Una caratteristica è la specificazione di un attributo. È una misurazione che rappresenta aspetti dei dati utili per evidenziare al fine di risolvere il problema considerato. Ad esempio, il colore può essere un attributo. "Il colore è blu" è una caratteristica estratta da un esempio. Possiamo vedere gli attributi come variabili casuali e le caratteristiche come i valori assunti da queste variabili (cioè, le caratteristiche sono "dati").

Il modello statistico da impiegare dipende in modo cruciale dalla scelta delle caratteristiche. Quindi, è utile considerare rappresentazioni alternative delle stesse misurazioni (cioè, caratteristiche diverse).

Le caratteristiche possono essere di due tipi principali:

- **Categoriche:** Un numero finito di valori discreti. Questi possono essere nominali, indicando che non c'è un ordine tra i valori, come i cognomi e i colori, o ordinali, indicando che esiste un ordine, come in un attributo che assume i valori basso, medio o alto.
- **Continue:** Comunemente, un sottoinsieme di numeri reali, dove c'è una differenza misurabile tra i valori possibili. Gli interi sono solitamente trattati come continui nei problemi pratici.

## ESEMPIO

Consideriamo il nostro esempio in cui vogliamo distinguere e-mail spam da non-spam. L'input del processo sono i messaggi e-mail, quindi dobbiamo trasformarli in vettori di caratteristiche  $\langle \#_1, \dots, \#_n \rangle$  con un processo di estrazione delle caratteristiche. Ovviamente, ci aspettiamo che le caratteristiche estratte siano utili per risolvere il nostro compito di determinare se un'e-mail è spam o non-spam.

Possiamo notare che le e-mail spam spesso includono errori ortografici e parole come "Compra", "occasione" e "\$100". Quindi, potremmo decidere di rappresentare ogni messaggio

e-mail con due numeri:

- Il conteggio degli errori ortografici.
- Il numero di volte che alcune parole o modelli specifici appaiono nel testo.

Una volta che i messaggi in input sono stati convertiti in vettori di caratteristiche, possono essere visti come vettori nello spazio bidimensionale (vedi figura). Questo è conveniente poiché vedremo che molti algoritmi di apprendimento automatico si basano su osservazioni geometriche.



## Tipi di Task

Rappresentare le tipologie di problemi che le macchine, altrimenti l'apprendimento automatico, cercano di risolvere

## Classificazione

Input e quale delle K categorie appartiene?

Algoritmo di apprendimento i solitamente prende come input un insieme di esempi  $\{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$  dove  $x^{(i)} \in \mathbb{R}^N$  per  $\forall i$ , e una serie di etichette corrispondenti  $\{y^{(1)}, \dots, y^{(n)}\}$  dove  $y^{(i)} \in \{1, \dots, k\} \forall i$ . Queste etichette specificano a quali delle K categorie appartiene ciascun esempio.

Nel caso di classificazione binaria (spam o non spam)  $y^{(i)} \in \{0, 1\}$ .

Per risolvere questo test, dobbiamo di K) essere le parole di uno puoi dire:

$$h: \mathbb{R}^N \rightarrow \{1, \dots, k\} : y^{(i)} = h(x^{(i)})$$

$$h_0(x) = y^{(i)} \Rightarrow \text{Obiettivo} \Rightarrow \text{ottenere un classificatore}$$

esempio target

h prendendo che dato un input produce un output che rappresenta le diverse stesse.

## Regressione

Predire un valore numerico <sup>continuo</sup> in base a un certo input

Siccome che classificazione, l'output viene fornito da esempi di addestramento  $x \in \mathbb{R}^n$  e da output desiderati  $y \in \mathbb{R}$

Algoritmo di ML osisce la ricerca di una funzione  $h: \mathbb{R}^n \rightarrow \mathbb{R} : y^{(i)} = h(x^{(i)})$

Classificazione VS Regressione

Output = etichetta di classe  
de insieme finito

BS. regressione  $\Rightarrow$  codice n. uscente da un posto

Output = predizioni numeriche

## Tipi di Learning

### Supervised Learning

Nell'apprendimento supervisionato, l'algoritmo viene fornito con esempi di input e output desiderati. In questo caso, il task può essere risolto adattando una funzione ai dati, ovvero creando una funzione che restituisce quei valori. Esempi di task di apprendimento supervisionato sono la **classificazione** e la **regressione**.

Algoritmo guarda se input de output desiderato per APPRENDIMENTO o RICERCA fine di essi.

Per affrontare problemi di machine learning, inclusi quelli di apprendimento supervisionato, spesso ci si affida a **modelli statistici** che divengono tali in fase di apprendimento. Questi modelli sono spesso **parametrizzati** (indicati con  $\theta$ ), e l'obiettivo è stabilire quali sono i parametri che meglio approssimano la funzione ideale sconosciuta

Preparare i dati  $\rightarrow$  Preprocess

Normalizzare feature

Le fonti discutono sia il **KNN** (**K-Nearest Neighbors**) che i **GMM** (**Gaussian Mixture Models**) nel contesto dell'apprendimento supervisionato.

Per quanto riguarda il **KNN**:

- Viene menzionato come uno degli **algoritmi di apprendimento supervisionato**.
- La sua logica si basa sul **confronto di alcuni campioni sconosciuti rispetto al corpo dei dati**. L'idea è di **comparare i campioni più simili** secondo una certa metrica e, in base alle etichette dei vicini, assegnare un'etichetta al nuovo campione.
- La **difficoltà** principale con KNN risiede nel **capire qual è la metrica migliore per comparare i campioni** e definire la **distanza del nuovo campione da ognuno degli altri**.
- La scelta della **metrica** è cruciale nello spazio delle feature. La **metrica euclidea** è un esempio facile da utilizzare su campioni con componenti numeriche, ma potrebbe non essere adatta se lo spazio delle feature non riflette bene le relazioni tra i dati.
- **Metric learning** è menzionato come una tecnica che cerca di **trasformare i dati** o di trovare una rappresentazione dei dati affinché sia possibile utilizzare una metrica, come quella euclidea, in modo efficace con KNN.
- Nel contesto di KNN, si distinguono i **parametri** del modello dagli **iperparametri**. La **metrica** scelta rientra nei parametri dell'algoritmo che si vogliono apprendere (come accennato in metric learning), mentre il **numero di vicini da considerare (K)** è identificato come un **iperparametro** che deve essere gestito esternamente al processo di apprendimento dei parametri. Il valore di K (quanti campioni vicini considerare per la decisione) non viene appreso direttamente dai dati con lo stesso meccanismo dei parametri del modello.

Per quanto riguarda i **GMM** (**Gaussian Mixture Models**):

- Sono presentati come un altro esempio di **algoritmo supervisionato**.
- Un GMM è descritto come la **sommatoria di più distribuzioni Gaussiane**, ognuna con il proprio set di **parametri (media e varianza)** e un **peso** che ne determina l'importanza nella miscela.
- Le fonti indicano che nei GMM, i **due set di parametri (quelli delle Gaussiane e i pesi)** e gli **iperparametri** non si apprendono contemporaneamente.
- È importante notare che la fonte menziona i **modelli generativi**, tra cui il **K-Means** (erroneamente indicato come Kix) e i GMM, nel contesto dell'**apprendimento non supervisionato**. In questo contesto, l'obiettivo è **stimare come i dati si distribuiscono nello spazio** e modellare la popolazione dei dati, spesso per poter generare nuovi campioni simili a quelli osservati. Questo sembra in contraddizione con la precedente menzione dei GMM come algoritmi supervisionati.

Pertanto, dalle fonti emerge che sia KNN che GMM sono menzionati come algoritmi che possono essere utilizzati nell'apprendimento supervisionato. Tuttavia, i GMM sono anche descritti nel contesto dell'apprendimento non supervisionato come modelli generativi. Questa duplice menzione suggerisce che i GMM possono essere utilizzati sia in contesti supervisionati (ad esempio, per la classificazione probabilistica, dove le componenti Gaussiane rappresentano le classi) che non supervisionati (per il clustering o la stima della densità). La fonte non approfondisce i dettagli di come i GMM sarebbero specificamente applicati in uno scenario di apprendimento supervisionato.

In sintesi:

- **KNN** è un algoritmo di apprendimento supervisionato basato sul concetto di similarità tra istanze, dove la scelta della metrica di distanza e del numero di vicini K sono aspetti chiave.
- **GMM** sono menzionati come algoritmi supervisionati, ma anche come modelli generativi non supervisionati per la modellazione della distribuzione dei dati. La fonte non specifica come vengano utilizzati precisamente nel contesto supervisionato.

## Unsupervised Learning

Nell'apprendimento non supervisionato, l'algoritmo viene fornito solo con esempi di input, senza etichette o output desiderati (quindi solo  $\{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$ , dove  $x^{(j)} \in \mathbb{R}^d$  per ogni  $j$ ). Questi tipi di task si concentrano generalmente sul modellare la struttura dei dati. L'aggregazione è un esempio di apprendimento non supervisionato, in cui non vengono fornite informazioni aggiuntive oltre agli esempi.

Gli approcci supervisionati sono generalmente più facili da gestire, ma richiedono la presenza di etichette. Ottenere le etichette è spesso un problema costoso in termini di tempo, poiché richiede che le persone annotino manualmente i dati. Ad esempio, se vogliamo costruire un rilevatore di spam utilizzando un approccio supervisionato, abbiamo bisogno di qualcuno che contrassegni manualmente diverse email come 'spam' o 'non-spam'.

Le fonti discutono l'**apprendimento non supervisionato (unsupervised learning)** come uno dei paradigmi di apprendimento, in contrasto con l'apprendimento supervisionato.

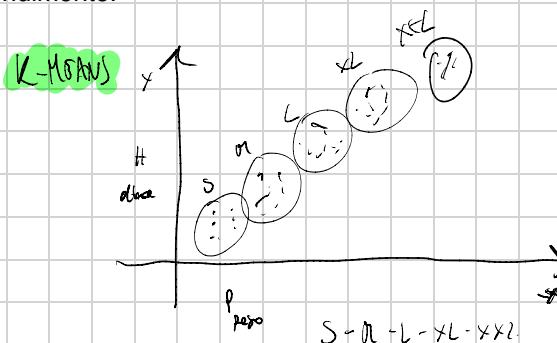
Nel **conto più ampio dei Tipi di Apprendimento**, la fonte menziona che oltre all'apprendimento supervisionato, esistono altri tipi di apprendimento che verranno trattati nel corso.

Successivamente, la fonte introduce specificamente l'apprendimento non supervisionato come un altro paradigma. Viene anche menzionato l'**apprendimento per rinforzo (reinforcement learning)** come un ulteriore tipo di apprendimento, basato su un processo di prova ed errore.

Per quanto riguarda l'**apprendimento non supervisionato** in dettaglio:

- La fonte spiega che in questo tipo di apprendimento, all'algoritmo vengono forniti **soltamente i dati di input X, senza alcuna etichetta o output associato**.
- L'**obiettivo** dell'apprendimento non supervisionato è quello di **stimare come i dati si distribuiscono nello spazio** in cui si trovano. In altre parole, si cerca di scoprire la struttura intrinseca dei dati senza una guida esterna fornita da etichette.
- Viene fornito un esempio concettuale in cui, avendo un unico valore come input ( $x$  con 1), l'apprendimento non supervisionato cerca di modellare la probabilità di osservare un dato in una certa area dello spazio, identificando le zone a maggiore o minore densità di dati.
- La fonte menziona i **modelli generativi** come un approccio spesso utilizzato nell'apprendimento non supervisionato. Questi modelli cercano di **modellare la distribuzione dei dati** in modo da poter generare nuovi campioni simili a quelli osservati.
- Un **esempio specifico di algoritmo di apprendimento non supervisionato** citato è il **K-Means (indicato anche come KX)**. Questo algoritmo cerca di **dividere lo spazio dei dati in un numero K di cluster o gruppi**. L'obiettivo è che gli elementi all'interno di ciascun gruppo siano simili tra loro in base alle loro caratteristiche (feature).
- La fonte illustra l'applicazione del K-Means con un esempio pratico: un'azienda che vuole regalare magliette di diverse taglie ai propri iscritti in base al peso e all'altezza registrati sulla carta fedeltà. Utilizzando il peso e l'altezza come input (senza conoscere a priori le taglie di magliette), l'algoritmo K-Means potrebbe raggruppare gli iscritti in un certo numero  $K$  di cluster, dove ogni cluster potrebbe essere associato a una taglia di maglietta. In questo caso, il **numero di cluster K** è identificato come un **parametro** dell'algoritmo K-Means.

In sintesi, le fonti presentano l'apprendimento non supervisionato come un paradigma distinto dall'apprendimento supervisionato, in cui l'algoritmo apprende solo dai dati di input per scoprire la loro struttura o distribuzione, fornendo l'algoritmo K-Means come esempio concreto. Viene anche accennato brevemente all'apprendimento per rinforzo come un ulteriore tipo di apprendimento.



## Reinforcement Learning

Le fonti menzionano l'**apprendimento per rinforzo (reinforcement learning)** come un tipo di apprendimento distinto dall'apprendimento supervisionato e non supervisionato.

Nel contesto più ampio dei **Tipi di Apprendimento**, la fonte indica che, oltre all'apprendimento supervisionato discusso precedentemente, e all'apprendimento non supervisionato, esiste anche l'apprendimento per rinforzo. Quest'ultimo si basa su un **processo di prova ed errore**.

Le caratteristiche principali dell'apprendimento per rinforzo, secondo le fonti, sono:

- **Processo di prova ed errore:** L'algoritmo prova diverse alternative e valuta se l'azione intrapresa è utile per raggiungere un obiettivo.
- **Ricompense e punizioni:** Se un'azione è considerata buona per il raggiungimento dell'obiettivo, l'algoritmo riceve una **regia positiva** (ricompensa). Se non è buona, riceve una **regia negativa** (punizione).
- **Apprendimento di policy:** L'obiettivo è quello di imparare delle **policy** che permettano all'algoritmo di prendere decisioni in autonomia data una situazione.
- **Valutazione a lungo termine:** Alla fine di un percorso di azioni, l'algoritmo può **sommare le regie** ricevute per valutare se quel percorso è ottimale rispetto ad altri.
- **Esempio di applicazione:** Un esempio fornito è quello dei **sistemi di navigazione autonoma**. Questi sistemi osservano l'ambiente e, in base a ciò che hanno appreso attraverso tentativi ed errori, decidono quale azione intraprendere (ad esempio, girare a sinistra) per raggiungere la destinazione.

In sintesi, le fonti presentano l'apprendimento per rinforzo come un paradigma in cui un agente impara a prendere decisioni interagendo con un ambiente attraverso prove ed errori, ricevendo feedback (ricompense o punizioni) per le sue azioni, con l'obiettivo di apprendere una strategia (policy) che massimizzi una ricompensa cumulativa nel tempo. Questo si differenzia dall'apprendimento supervisionato, in cui si apprendono mapping da input a output basandosi su dati etichettati, e dall'apprendimento non supervisionato, in cui si cerca di scoprire la struttura nascosta nei dati non etichettati.

## Valutazioni delle Performance

Le fonti discutono in maniera significativa la **Valutazione della Performance** nel contesto più ampio del **Machine Learning**, sottolineando la sua importanza per determinare l'efficacia degli algoritmi.

Nel contesto dei **tipi di task** affrontati nel machine learning, le fonti distinguono principalmente tra **classificazione e regressione**, e indicano come la valutazione della performance vari in base a questi:

- **Classificazione:** In questo scenario, l'obiettivo è assegnare un input a una classe specifica. La performance è comunemente misurata utilizzando l'**accuratezza**, definita come la **percentuale di risposte corrette** fornite dal modello. Questo si calcola confrontando le **etichette reali (ground truth)** con le **etichette stimate (output del modello)**. Ad esempio, se un modello classifica correttamente quattro email su cinque, la sua accuratezza è dello 0.8 o 80%.
- **Regressione:** Qui, l'obiettivo è prevedere un valore numerico continuo. Invece dell'accuratezza, la performance è valutata tramite **misure di errore** che quantificano la differenza tra i valori reali e quelli predetti. Un esempio menzionato è l'**errore quadratico (Squared Error)**, e l'obiettivo è che questo errore sia il più basso possibile.

Le fonti enfatizzano un concetto cruciale nella valutazione: la **generalizzazione**. È fondamentale valutare la performance di un modello su **dati che non sono stati utilizzati durante la fase di apprendimento (training)**. Utilizzare i dati di training per la valutazione può fornire risultati ingannevoli, in quanto il modello potrebbe aver semplicemente memorizzato i dati di training senza essere in grado di fare previsioni accurate su nuovi dati.

Per questo motivo, i dati vengono tipicamente suddivisi in diversi set:

- **Dataset di training:** Utilizzato per **apprendere i parametri** del modello.
- **Dataset di test:** Utilizzato per **valutare la performance finale** del modello su dati mai visti e stimare la sua capacità di generalizzazione.
- **Dataset di validazione:** Introdotto quando è necessario **ottimizzare gli iperparametri** di un modello o per **confrontare diversi modelli**. La performance su questo set guida la selezione degli iperparametri o del modello migliore, ma non rappresenta la valutazione finale.

Le fonti spiegano che una **funzione di performance** confronta l'insieme delle etichette reali con quelle stimate dal modello. Il valore di questa funzione indica la bontà della performance del modello. Un **valore alto** è generalmente associato a una buona performance (come nel caso dell'accuratezza), mentre un **valore basso** indica una performance scarsa (come nel caso dell'errore), ma l'interpretazione dipende dalla **forma specifica della funzione matematica**. Ad esempio, l'accuratezza è una misura in cui un valore crescente indica una performance migliore, mentre l'errore è una misura in cui un valore crescente indica una performance peggiore. È anche menzionato che una misura di errore può essere calcolata come 1 meno una misura di accuratezza.

Infine, le fonti sottolineano che la valutazione della performance avviene generalmente sull'**intero set di test**, e sono necessarie formule per calcolare le misure di performance considerando tutti i campioni nel set. Questo fornisce una valutazione complessiva di quanto bene il modello generalizza a nuovi dati.

## Espiazione e Dati

Le fonti sottolineano che l'**esperienza**, nel contesto del Machine Learning, è fondamentalmente costituita dai **dati** o **dataset** utilizzati per l'apprendimento di un algoritmo.

Nel paradigma del Machine Learning, in contrasto con la programmazione tradizionale in cui si definiscono esplicitamente i passaggi logici per risolvere un problema, si cerca di far apprendere alle macchine come risolvere uno specifico problema a partire dai dati. Questo cambiamento di paradigma è necessario soprattutto per classi di problemi caratterizzati da alta incertezza, dove non è facile definire una logica precisa o fare assunzioni complete per la risoluzione, come nel caso della classificazione di immagini o di email come spam.

La definizione di "apprendimento" fornita nelle fonti evidenzia tre componenti chiave: un computer apprende da un'**esperienza** rispetto a uno specifico **task** e a una **misura di performance**, se la performance nel task migliora con l'esperienza. In questo contesto, l'**esperienza** è rappresentata dai **dati**.

Le fonti descrivono in dettaglio la natura di questi dati, specialmente nel contesto dell'apprendimento supervisionato:

- L'**esperienza** si manifesta come un **insieme di dati**, dove ogni dato è chiamato **esempio**.
- Nel caso della classificazione di email, un **dato** (esempio) è composto dal **corpo della mail** e dall'**etichetta associata** (spam o non spam). Il corpo della mail rappresenta l'**input (X)**, mentre l'etichetta rappresenta l'**output desiderato (y)**.
- Generalmente, un **esempio (X)** è espresso come una **collezione di valori o misure quantificabili**, formando un **vettore** di caratteristiche (feature vector). Ogni componente di questo vettore è un **attributo**, e il valore assunto da questo attributo è la **feature**. Gli attributi possono essere di diverso tipo (categorici nominali, categorici ordinali, continui), il che influenza la scelta dei modelli statistici.
- Un **dataset** può essere organizzato come una **matrice di design**, dove le **righe** corrispondono agli **esempi** e le **colonne** corrispondono agli **attributi** (feature).

Le fonti sottolineano anche l'importanza della **rappresentazione dei dati** attraverso l'**estrazione di feature**. Il dato grezzo (come il contenuto di un'email) potrebbe non essere direttamente utilizzabile dall'algoritmo. È necessario un processo di **estrazione di feature** per trovare una **rappresentazione** che contenga le **caratteristiche salienti** per il task specifico. Questa trasformazione dei dati fa parte del modello.

Nel contesto dell'apprendimento supervisionato, l'algoritmo di learning riceve in input sia i dati (X) che le etichette associate (y). L'obiettivo è apprendere una funzione (modello) che possa mappare nuovi input alle corrispondenti uscite.

Per valutare se un modello ha appreso correttamente e se è in grado di generalizzare a nuovi dati, è fondamentale utilizzare **dataset distinti** da quelli utilizzati per l'apprendimento:

- **Dataset di training:** Utilizzato per **apprendere i parametri** del modello.
- **Dataset di test:** Utilizzato per **valutare la performance finale** del modello su dati mai visti e stimare la sua capacità di **generalizzazione**.
- **Dataset di validazione:** Utilizzato per **ottimizzare gli iperparametri** del modello o per **confrontare diversi modelli**, fornendo una stima della performance che guida la scelta, ma che non è la valutazione finale.

Infine, le fonti menzionano che spesso è necessario applicare delle **trasformazioni** ai dati, come la **normalizzazione** (Min-Max scaling, Z-score standardization), per portare le diverse feature a vivere in uno stesso spazio o per centrare e scalare i dati. Queste trasformazioni vengono apprese dai dati di training e poi applicate anche ai dati di test e a qualsiasi nuovo dato. Queste trasformazioni diventano parte integrante del modello.

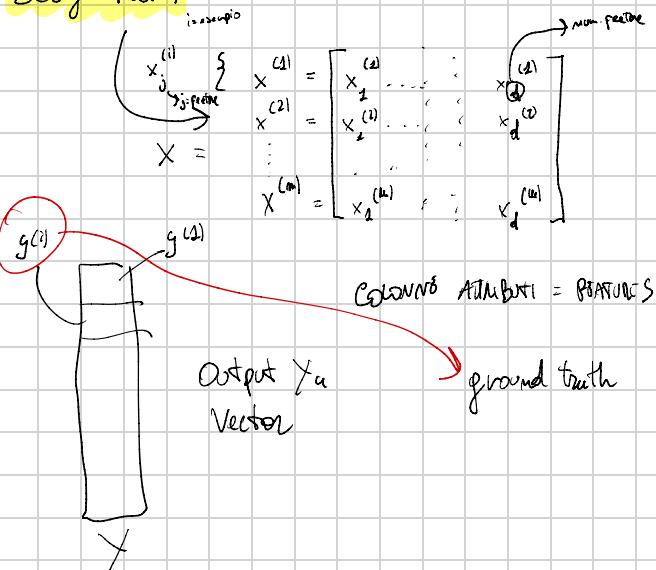
In sintesi, l'**esperienza** nel Machine Learning è concretamente rappresentata dai **dati (dataset)**, che devono essere preparati, rappresentati in modo efficace attraverso l'estrazione di feature e suddivisi in modo appropriato per l'apprendimento, la validazione (se necessaria) e la valutazione della performance e della capacità di generalizzazione del modello.

Set di dati  $\rightarrow$  Design Matrix

$$X \in \mathbb{R}^{m \times d}$$

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

$$A \in \{1, \dots, k\}$$



### La Design Matrix nel Machine Learning

Le fonti descrivono in dettaglio come l'esperienza, rappresentata dai dati (dataset), viene strutturata e organizzata per essere utilizzata negli algoritmi di Machine Learning, introducendo il concetto di Design Matrix (X).

### L'Esperienza nei Dati

Nel contesto dell'apprendimento automatico, l'esperienza da cui un computer apprende e' costituita da un insieme di dati. Ogni dato e' chiamato esempio e si presenta generalmente come una collezione di valori o misure quantificabili, formando un vettore di caratteristiche (feature vector).

- Ogni componente di questo vettore e' un attributo.
- Il valore assunto dall'attributo e' la feature.

### Struttura della Design Matrix (X)

Un dataset puo' essere organizzato come una matrice di design (X). Le fonti spiegano chiaramente la sua struttura:

- **Righe:** Corrispondono ai **singoli esempi** (o record) presenti nel dataset. **Ogni riga rappresenta un'osservazione o un punto dati su cui si basa l'apprendimento.**  
Esempio: Nella classificazione di email, ogni riga potrebbe rappresentare una singola email.
- **Colonne:** Corrispondono agli **attributi** (o feature). Ogni colonna rappresenta una specifica caratteristica misurabile degli esempi.  
Esempio: Nel contesto di email spam, le colonne potrebbero includere la frequenza di certi caratteri, la presenza di errori ortografici o di parole chiave.

### Formalizzazione Matematica

La fonte formalizza ulteriormente la struttura della matrice:

- Se il dataset contiene n esempi, la matrice X avra' n righe.
- Se ogni esempio e' descritto da d attributi (feature), la matrice X avra' d colonne.
- La matrice X appartiene a  $R^{n \times d}$ , dove R indica i numeri reali, n e' il numero di esempi e d e' la dimensione (numero di feature) di ogni esempio.
- Un elemento  $X_{i,j}$  rappresenta la caratteristica j-esima dell'esempio i-esimo del dataset.

### Rappresentazione ed Estrazione delle Feature

Le fonti sottolineano che la rappresentazione dei dati attraverso l'estrazione di feature e' un passaggio cruciale. I dati grezzi, come il testo di un'email, devono essere trasformati in un formato numerico e significativo per l'algoritmo. Le feature estratte diventano le colonne della matrice di design e fanno parte integrante del modello.

### Apprendimento Supervisionato: La Matrice di Output (Y)

Nel contesto dell'apprendimento supervisionato, oltre alla matrice di input X, si ha spesso una matrice di output Y (o un vettore, nel caso di un singolo output per esempio), che contiene le etichette o i valori target associati a ciascun esempio in X.

- Classificazione: Per K classi, Y potrebbe appartenere a  $R^{m \times k}$ , dove m e' il numero di esempi e K e' il numero di classi. In alternativa, puo' essere un vettore di lunghezza m, dove ogni elemento indica la classe corrispondente.
- Regressione: Y conterrà i valori numerici da prevedere.

### Trasformazioni della Design Matrix

Le fonti evidenziano che spesso e' necessario applicare trasformazioni alla matrice di design, come:

- Normalizzazione (es. Min-Max scaling, Z-score standardization), per garantire che le feature operino su scale comparabili.

Queste trasformazioni, apprese dai dati di training, vengono applicate anche ai dati di test e validazione, diventando parte integrante del modello.

Matrice delle etichette → ruolo cruciale!

Apprendendo parametri modello → Durante training, adattamento di ML mira a trovare una funzione  $h_\theta$  in grado di mappare gli input X alle uscite Y  
Ottimizzazione → Minimizzare diff. etichette reale y e predette  $\hat{y}$

Evaluation → Una volta addestrato il modello, è necessario valutarne la capacità di generalizzare a nuovi dati non visti. In questa fase, il modello viene utilizzato per predire le etichette ( $\hat{y}$ ) per un nuovo set di dati di input (dataset di test). Le etichette reali (Y) del dataset di test vengono poi confrontate con le etichette predette ( $\hat{y}$ ) utilizzando una funzione di performance (come l'accuratezza per la classificazione o l'errore quadratico medio per la regressione) per quantificare quanto bene il modello ha appreso il task. Le fonti sottolineano l'importanza di valutare la performance su dati di test separati dai dati di training per evitare problemi di "memorizzazione" e sovrallenamento.

## Learning / Train del Modello

Prende in input:

Dataset D

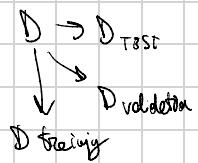
Modello (Parametri)

Misure di Performance

Output

Set di parametri che produce la miglior performance su D

Evaluation → Generalizzare



## Concetto di suddivisione di un dataset

Il concetto di suddivisione di un dataset  $D$  in *training*, *validation* e *test set* è fondamentale nell'apprendimento automatico per costruire modelli robusti e generalizzabili.

- **Dataset  $D$ :** È l'insieme totale dei dati disponibili, composto da esempi (input e output attesi).

### Suddivisione:

- **Training set ( $D_{\text{train}}$ ):** Una porzione del dataset (es. 60-80%) usata per addestrare il modello, cioè per ottimizzare i parametri del modello  $h$  (es. pesi di una rete neurale). Il modello “impara” dai dati di training adattando i parametri per minimizzare l’errore di previsione.
- **Validation set ( $D_{\text{validation}}$ ):** Una porzione separata (es. 10-20%) usata per ottimizzare gli iperparametri del modello (es. learning rate, numero di strati, ecc.). Si valuta il modello su questo set durante l’addestramento per scegliere la configurazione migliore senza toccare il test set.
- **Test set ( $D_{\text{test}}$ ):** Una porzione finale (es. 10-20%) usata solo alla fine per validare le prestazioni del modello. Serve a stimare quanto bene il modello generalizza su dati mai visti, dando una misura realistica della sua efficacia.

### Perché questa suddivisione?

- **Training:** Serve per far imparare al modello i pattern nei dati.
- **Validation:** Permette di regolare gli iperparametri e prevenire l’overfitting (quando il modello si adatta troppo ai dati di training e non generalizza).
- **Test:** Fornisce una valutazione imparziale delle prestazioni finali, simulando come il modello si comporterà su dati nuovi.

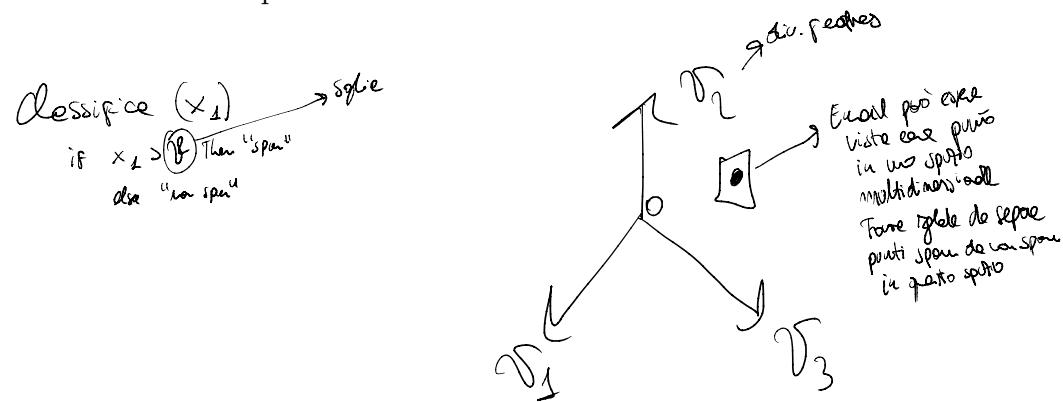
### Flusso di lavoro:

1. Alleni il modello su  $D_{\text{train}}$  aggiustando i parametri  $h$ .
2. Valuti diverse configurazioni su  $D_{\text{validation}}$  per scegliere gli iperparametri ottimali.
3. Una volta scelto il modello finale, lo testi su  $D_{\text{test}}$  per verificarne l’accuratezza.

**Esempio pratico:** Se hai un dataset di immagini di gatti e cani:

- $D_{\text{train}}$ : Usato per insegnare al modello a distinguere gatti da cani.
- $D_{\text{validation}}$ : Usato per decidere, ad esempio, quante epoche di addestramento o quale architettura della rete usare.
- $D_{\text{test}}$ : Usato per verificare se il modello riconosce correttamente gatti e cani su immagini mai viste.

I dati devono essere mescolati casualmente prima della suddivisione per evitare bias, e le proporzioni (es. 70-15-15) possono variare in base al problema.



Variables

↳ videos are vector class

INPUT  $x^{(i)} = [x_1^{(i)}, \dots, x_d^{(i)}]^T$

$$X^{(i)} = \begin{bmatrix} x_{1,1}^{(i)} & \dots & x_{1,n}^{(i)} \\ \vdots & & \vdots \\ x_{c,1}^{(i)} & \dots & x_{c,n}^{(i)} \end{bmatrix}$$

inner scale of pix

$$X^{(i)} = \begin{bmatrix} 1 & \dots & 1 \\ \vdots & & \vdots \\ 1 & \dots & 1 \end{bmatrix}$$

Tensor inner  $l \times B \rightarrow 3$  card:  $m \times n \Rightarrow m \times n \times 3$

same inner  $l \times B \rightarrow 3$  card:  $m \times n \Rightarrow m \times n \times 3 \times t$   $\rightarrow$  istanti di tempo

$$\mathcal{V}^i = [\mathcal{V}_0, \dots, \mathcal{V}_d]^T \rightarrow \text{feature models}$$

$$X^{(i)} = [x_0^{(i)}, x_1^{(i)}, \dots, x_d^{(i)}]^T \Rightarrow \begin{bmatrix} x_0^{(i)} & \dots \\ x_1^{(i)} & \dots \\ \vdots & \dots \\ x_d^{(i)} & \dots \end{bmatrix}$$

$\uparrow$   $\rightarrow s_0$

## Contesto generale

Nel contesto della regressione lineare o delle reti neurali, ogni esempio di input viene rappresentato come un vettore  $\mathbf{X}^{(i)}$ , dove  $i$  indica il  $i$ -esimo esempio nel dataset. Solitamente, questo vettore contiene le caratteristiche (features) osservate per quell'esempio. Ad esempio:

$$\mathbf{X}^{(i)} = \left[ x_1^{(i)}, x_2^{(i)}, \dots, x_d^{(i)} \right]^T$$

Qui:

- $x_j^{(i)}$  è il valore della  $j$ -esima feature per l'esempio  $i$ .
- $d$  è il numero totale di features.

## Problema dell'intercetta (bias)

In molte formule matematiche, ad esempio in una regressione lineare o in un modello lineare generico, si desidera includere un termine costante (intercetta) nel modello. Questo termine permette al modello di spostarsi verticalmente lungo l'asse  $y$ , senza essere vincolato a passare attraverso l'origine  $(0, 0)$ .

Matematicamente, se stiamo lavorando con una funzione lineare del tipo:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d$$

il termine  $\theta_0$  è l'intercetta (bias). Per rendere questa formula più compatta e uniforme, possiamo riscriverla utilizzando un prodotto scalare tra i parametri  $\boldsymbol{\theta}$  e le features  $\mathbf{X}$ :

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d = \sum_{j=0}^d \theta_j x_j$$

Per farlo, dobbiamo estendere il vettore  $\mathbf{X}^{(i)}$  aggiungendo una nuova feature  $x_0^{(i)}$  che vale sempre 1. In questo modo, il termine  $\theta_0$  può essere trattato come un parametro normale, moltiplicato per  $x_0^{(i)} = 1$ .

## Estensione del vettore di input

La nota specifica:

$$\mathbf{X}^{(i)} = \left[ x_0^{(i)} = 1, x_1^{(i)}, \dots, x_d^{(i)} \right]^T$$

Questo significa che:

- Al vettore di input  $\mathbf{X}^{(i)}$  viene aggiunta una nuova feature  $x_0^{(i)}$  che vale sempre 1.
- Il nuovo vettore diventa di dimensione  $d + 1$ , dove la prima componente è sempre 1.

Ad esempio, se l'input originale era:

$$\mathbf{X}^{(i)} = \left[ x_1^{(i)}, x_2^{(i)}, x_3^{(i)} \right]^T$$

Dopo l'estensione, diventa:

$$\mathbf{X}^{(i)} = \left[ 1, x_1^{(i)}, x_2^{(i)}, x_3^{(i)} \right]^T$$

## Motivo dell'estensione

L'estensione serve principalmente per due ragioni:

- **Semplificazione delle formule:** Consentendo di trattare l'intercetta  $\theta_0$  come un parametro normale, possiamo scrivere la predizione del modello come un semplice prodotto scalare:

$$\hat{y}^{(i)} = \boldsymbol{\theta}^T \mathbf{X}^{(i)}$$

Dove:

$$\boldsymbol{\theta} = [\theta_0, \theta_1, \dots, \theta_d]^T \quad \text{è il vettore dei parametri.}$$

$$\mathbf{X}^{(i)} = \left[ 1, x_1^{(i)}, \dots, x_d^{(i)} \right]^T \quad \text{è il vettore esteso.}$$

- **Unificazione del calcolo:** Con questa notazione, tutte le operazioni (ad esempio, il calcolo del gradiente durante l'ottimizzazione) possono essere gestite in modo uniforme, senza bisogno di distinguere esplicitamente il termine  $\theta_0$ .

## Associazione di $x_0^{(i)} = 1$ a $\theta_0$

La nota evidenzia anche: “ASSOCIATE TO  $\theta_0$ .”

Ciò significa che il termine  $x_0^{(i)} = 1$  è progettato appositamente per associarsi al parametro  $\theta_0$ , che rappresenta l’intercetta del modello. Quando si effettua il prodotto scalare  $\boldsymbol{\theta}^T \mathbf{X}^{(i)}$ , il termine  $\theta_0 \cdot x_0^{(i)}$  diventa semplicemente  $\theta_0 \cdot 1 = \theta_0$ , mantenendo il ruolo di intercetta.

## Esempio pratico

Supponiamo di avere un esempio con tre features  $x_1, x_2, x_3$ :

$$\mathbf{X}^{(i)} = \left[ x_1^{(i)}, x_2^{(i)}, x_3^{(i)} \right]^T = [2, 3, 4]^T$$

Dopo l'estensione:

$$\mathbf{X}^{(i)} = \left[ 1, x_1^{(i)}, x_2^{(i)}, x_3^{(i)} \right]^T = [1, 2, 3, 4]^T$$

Il vettore dei parametri  $\boldsymbol{\theta}$  sarà:

$$\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, \theta_3]^T$$

La predizione del modello sarà:

$$\hat{y}^{(i)} = \boldsymbol{\theta}^T \mathbf{X}^{(i)} = \theta_0 \cdot 1 + \theta_1 \cdot x_1^{(i)} + \theta_2 \cdot x_2^{(i)} + \theta_3 \cdot x_3^{(i)}$$

Che si riduce a:

$$\hat{y}^{(i)} = \theta_0 + \theta_1 \cdot 2 + \theta_2 \cdot 3 + \theta_3 \cdot 4$$

## Conclusioni

L’aggiunta di  $x_0^{(i)} = 1$  all’input  $\mathbf{X}^{(i)}$  è una convenzione comune che semplifica le formule matematiche e permette di trattare l’intercetta  $\theta_0$  come un parametro normale. Questa tecnica è particolarmente utile nei modelli lineari e nelle reti neurali, dove si desidera mantenere una notazione uniforme e compatta.

# Pre-processing dei Dati (Feature Scaling) nel Machine Learning

Le fonti discutono il **pre-processing dei dati (Feature Scaling)**, in particolare la **normalizzazione**, come una fase cruciale nel contesto più ampio del **Machine Learning**.

Le fonti evidenziano che le diverse **feature** di un dataset possono esistere in **spazi con range di valori differenti**. Questa disparità di scale può influenzare il calcolo della distanza tra i punti nello spazio delle feature. Per affrontare questo problema, le fonti descrivono tecniche di **normalizzazione** che mirano a portare i valori delle feature in uno **stesso spazio** o intervallo.

## Tecniche di Normalizzazione

Nello specifico, vengono menzionate due tecniche di normalizzazione:

- **Scaling Min-Max:** Questa tecnica **scala i valori di una feature per rientrare nell'intervallo 0 e 1**. La formula fornita è:

$$X'_j = \frac{X_j - X_{jmin}}{X_{jmax} - X_{jmin}}$$

dove  $X_j$  è il valore originale della  $j$ -esima componente,  $X_{jmin}$  è il valore minimo osservato per quella componente nel dataset di training, e  $X_{jmax}$  è il valore massimo osservato per quella componente nel dataset di training. Questa trasformazione viene applicata a **ogni colonna (feature)** dei dati di training.

- **Z-Score Standardization:** Questa tecnica **trasforma i valori di una feature in modo che abbiano una media di zero e una deviazione standard di uno**. La formula fornita è:

$$x'_j = \frac{x_j - \mu_j}{\sigma_j}$$

dove  $x_j$  è il valore originale della  $j$ -esima componente,  $\mu_j$  è la **media** dei valori di quella componente nel dataset di training, e  $\sigma_j$  è la **deviazione standard** dei valori di quella componente nel dataset di training. Questa trasformazione **centra i dati intorno allo zero** e li scala in base alla loro dispersione.

## Importanza della Normalizzazione nel Processo

Le fonti sottolineano un aspetto fondamentale: queste trasformazioni di normalizzazione vengono apprese e applicate sui **dati di training**. Successivamente, **la stessa trasformazione deve essere applicata a qualsiasi nuovo campione di test** che si desidera processare con il modello appreso. I valori di  $X_{j\min}$ ,  $X_{j\max}$ ,  $\mu_j$ , e  $\sigma_j$  calcolati sul dataset di training diventano **parte del modello**. Pertanto, il processo di trasformazione dei dati è considerato un **componente integrante del modello di machine learning**.

Nel contesto più ampio del Machine Learning, il pre-processing dei dati, inclusa la normalizzazione, è una fase che precede l'addestramento del modello. Prima della normalizzazione, può avvenire l'**estrazione delle feature**, un processo che trasforma l'input grezzo in una rappresentazione più utile per il modello. La normalizzazione agisce quindi sui dati trasformati (le feature) per garantire che tutte contribuiscano equamente all'apprendimento.

## Conclusione

In sintesi, le fonti presentano la normalizzazione come una **tappa essenziale del pre-processing dei dati (Feature Scaling)** nel Machine Learning. Essa serve a **portare le feature su una scala comparabile**, il che è cruciale per il corretto funzionamento di molti algoritmi e per assicurare che il modello generalizzi bene a dati non visti. Le tecniche di normalizzazione e i loro parametri derivati dai dati di training sono considerati **parte integrante del modello finale**.