Famiglia del Percettrone

Ivan Maieli

4 giugno 2023

Indice

Capitolo 1

Introduzione

1.1 Cosa ci aspetta

Ci troviamo in periodo storico dove l'interpretazione dei dati rappresenta una disciplina fondamentale, ed è proprio per questo motivo che esperti del settore come data scientist e data analysts fanno leva sul famoso apprendimento automatico. Al giorno d'oggi, le grandi aziende internazionali si trovano a pulire, processare e manipolare un'enorme quantità di dati, inimmaginabile per la nostra mente.
Questo piccolo testo non tratta dell'etica dietro la raccolta di questi dati, argomento molto discusso e che, con molta calma e dedizione, meriterebbe un approfondimento separato.
È importante comprendere che, quello che andremo a vedere, non rappresenta in alcun modo una guida completa su questo argomento, bensì una semplicissima (perchè semplicissima è) panoramica sull'uso di questi spettacolari strumenti, distanti e concettualmente separati da come vediamo le intelligenze artificiali in questi giorni. Avremo modo di discutere dei diversi tipi di dati, dei diversi tipi di algoritmi e della rappresentazione in linguaggio C dei precursori, in modo da comprendere il pensiero logico dietro questa branca dell'informatica e della psicologia.
Vedremo come, in un certo senso, il legame tra il concetto umano di intelligenza e quello artificiale siano dapprima vicini e, successivamente, distanti.
Non è stato semplice avanzare in questa materia restando vicini alla sola nozione di intelligenza senza scomodare la matematica, l'unica lingua in grado di descrivere perfettamente la natura e l'universo. Leonardo da Vinci, primo

uomo a studiare un modo per permettere all'uomo di volare, pensò che la chiave del successo in questo esperimento fosse lo studio delle ali e del volo degli uccelli, osservazione indubbiamente corretta da un certo punto di vista ma che, dall'altro, non permise di raggiungere l'obiettivo. Oggi sappiamo volare e, anche se un aereo rispecchia molto un volatile, non segue gli stessi principi del volo.
È successa una cosa analoga a questa nel tentativo di creare algoritmi di IA, ormai poco vicini al nostro modo di pensare e apprendere.

1.2 Machine Learning

La maggior parte delle persone, quando sente le parole "Machine Learning", pensa ad un robot: a volte con riferimenti a Terminator, altre volte ad altri robot futuristici. Non negando il fatto che possano esserci coinvolti robot, tenderei a negare invece la visione futuristica, ormai sfatata da tempo: è chiaro che non si può parlare più di futuro in una materia più che mai attuale.

Cos'è il Machine Learning? Esistono diverse risposte a questa domanda, alcune particolarmente ingegneristiche, altre meno, ma tutte comunque fondamentali per aggiungere un pezzo al puzzle complessivo.

Alcune di queste potrebbero essere:

"Il Machine Learning è definito come la capacità di una macchina di apprendere, adattarsi e risolvere automaticamente problemi complessi con precisione e prestazioni crescenti."

"Il Machine Learning è una branca dell'intelligenza artificiale (AI) e dell'informatica che si concentra sull'uso di dati e algoritmi per imitare il modo in cui gli esseri umani imparano, migliorando gradualmente la loro precisione."

-IBM

"Il Machine Learning è il processo di utilizzo di modelli matematici di dati per aiutare un computer ad apprendere senza istruzioni dirette."

-Microsoft

"Si dice che un programma per computer impara dall'esperienza E rispetto a un compito T e a una misura di prestazione P, se la sua prestazione su T, misurata da P, migliora con l'esperienza E."

-Tom Mitchell

Tentando di mettere insieme i pezzi, il Machine Learning sfrutta innegabilmente la disciplina informatica e i suoi principi d'automazione, che permettono di iterare procedimenti altrimenti infattibili per un qualunque essere umano. Viene svolto uno studio accurato della statistica, dell'algebra lineare, dell'analisi matematica e della programmazione.

1.2.1 I diversi tipi

Questo potente strumento si mostra duttile in parecchie situazioni diverse: nei compiti di classificazione, nei compiti di regressione, nei compiti di raggruppamento o clustering e nei compiti di riduzione della dimensionalità. Il Machine Learning si divide in tre principali categorie:

- Supervised Learning (Apprendimento Supervisionato)
- Unsupervised Learning (Apprendimento Non Supervisionato)
- Reinforcement Learning (Apprendimento di Rafforzamento)

1.2.2 Supervised Learning

Sfruttiamo il Supervised Learning quando addestriamo il nostro algoritmo fornendo un set di dati con le soluzioni, chiamate *label* o *etichette*. Tutte le colonne diverse da quella delle etichette vengono dette *feature*, e rappresentano le caratteristiche dell'oggetto che andremo a definire e studiare.

Questo è un esempio (AND) della possibile struttura:

x_1 - feature 1	x_2 - feature 2	y - label
0	0	0
0	1	0
1	0	0
1	1	1

Comprendiamo come la possibilità di avere i risultati sia importantissima. Questa ci permette di avere informazioni sugli errori commessi in modo da agire di conseguenza al fine di ridurli.
Uno dei primi esempi di Supervised Learning è l'email spam o non spam: addestriamo il nostro algoritmo su centinaia e centinaia di email già classificate come spam o non spam, in modo che l'algoritmo riesca a scindere le due classi. Una volta addestrato, il nostro obiettivo sarà permettere la generalizzazione, ovvero la capacità di fornire risultati accurati su campioni mai visti. In sostanza forniamo noi le *features* del nostro campione, ed esso (l'algoritmo) dovrà essere in grado di produrre un risultato attendibile.

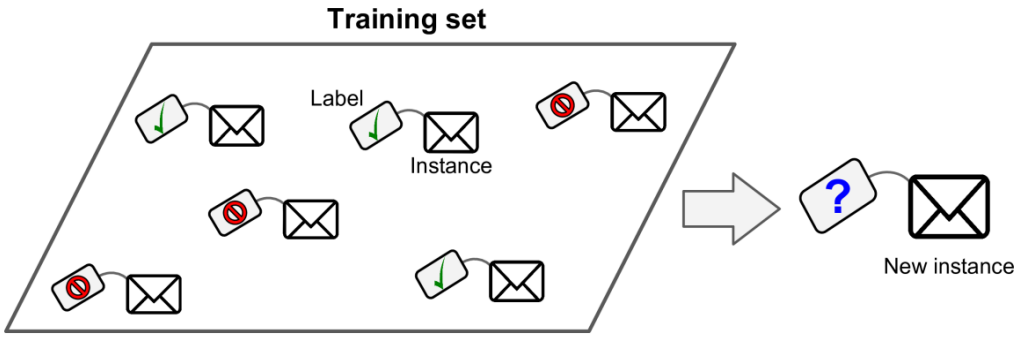

Figura 1.1: Esempio del funzionamento della classificazione.

La struttura dei dati, il numero di *features* e il numero di campioni sono fondamentali per il corretto addestramento dell'algoritmo. La cultura occidentale ha forgiato tutti i suoi seguaci sviluppatori secondo la logica che il dataset rappresenti la sola discriminante, ma non è così.
Spesso si trascura fin troppo l'algoritmo che, come vedremo nei capitoli successivi, sfrutta moltissimi parametri in grado di ottimizzare il modello.

Oltre la classificazione, ossia l'appartenenza di un campione ad una determinata classe, abbiamo la regressione, ovvero la ricerca di un target scalare. Un esempio comune potrebbe essere la stima di valori, come il prezzo di una macchina o di una casa, dove le *features* potrebbero essere le miglia percorse, la casa produttrice, la cilindrata e così via.

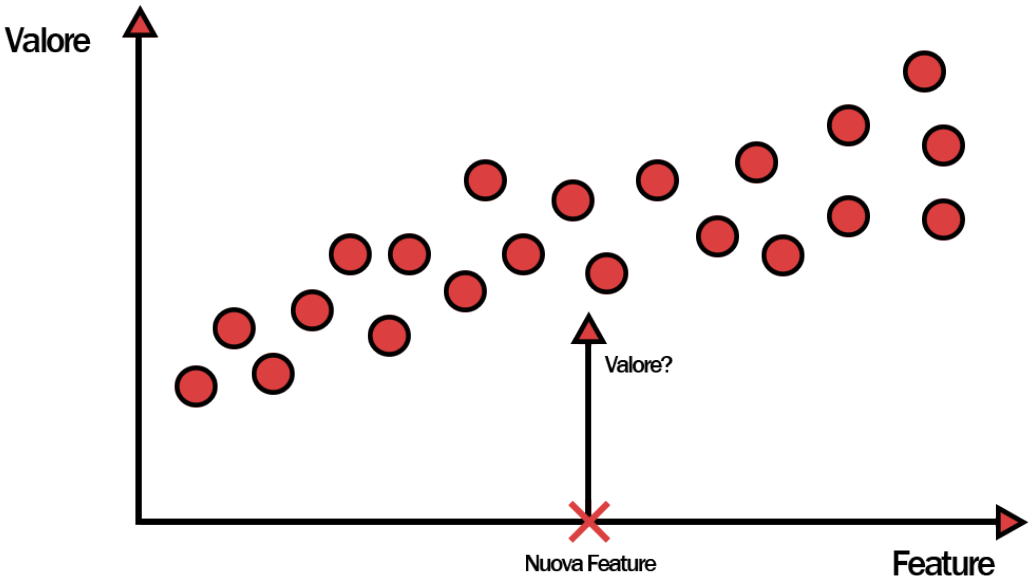

Figura 1.2: Esempio del funzionamento della regressione.

Abbiamo a nostra disposizione parecchi algoritmi diversi, con altrettanti punti di forza e punti deboli. Alcuni di essi si prestano maggiormente per la classificazione supervisionata, altri per la regressione supervisionata.
I più famosi sono:

- Perceptron (Non utilizzato)
- k-Nearest Neighbors
- Regressione Lineare
- Regressione Logistica
- Support Vector Machines (SVM)
- Decision Trees e Random Forests
- Reti Neurali

1.2.3 Unsupervised Learning

Come suggerisce il nome, l'Unsupervised Learning non ha la guida delle *labels*, quindi il suo compito non sarà quello di fornire output numerici.
I suoi algoritmi si dividono in:

- Clustering
  - K-Means
  - DBSCAN
  - Hierarchical Cluster Analysis (HCA)
- Dimensionality Reduction
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - t-distributed Stochastic Neighbor Embedding (t-SNE)

Un *cluster* è corrispondente ad un gruppo quindi, il *clustering*, è un algoritmo in grado di svolgere un raggruppamento.
Questo tipo di algoritmi vengono utilizzati dalle aziende per trovare relazioni tra i clienti e, di conseguenza, promuovere in maniera accurata determinati prodotti.

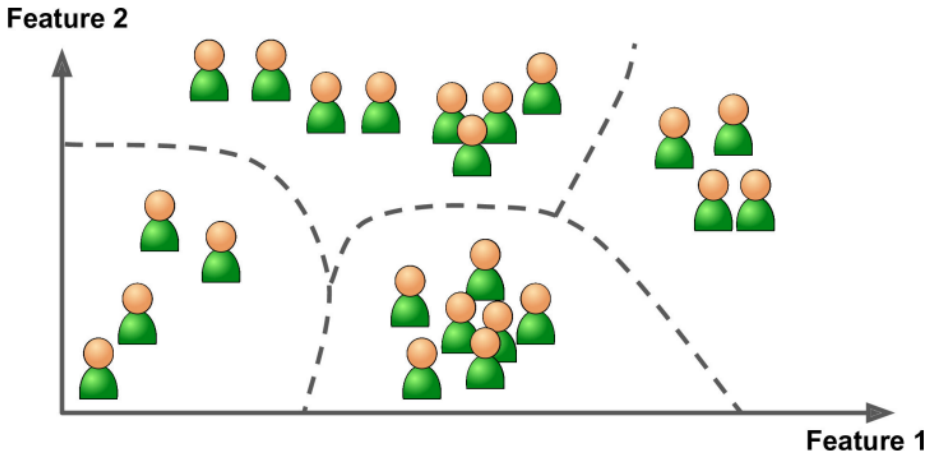

Figura 1.3: Esempio del funzionamento del clustering.

La *Dimensionality Reduction* invece ha un compito un pò più statistico e, prestatemi il concetto, in grado di semplificare il dataset.
Osservando un dataset come array di *feature*, ovvero altri vettori, potremo dire che il dataset ha tante dimensioni tante quante sono le *features*. Per ovvi motivi legati prevalentemente alla visualizzazione e alla complessità dei calcoli, si è cercato di sviluppare degli algoritmi in grado di ridurre le dimensioni associando le features in base alla correlazione.
Ritornando all'esempio precedente dell'auto, troveremo, avventurandoci tra i dati, che le miglia percorse saranno fortemente correlate all'età, di conseguenza l'algoritmo le unirà (merge) tra loro. Questo permetterà la semplificazione della visualizzazione e del calcolo computazionale.

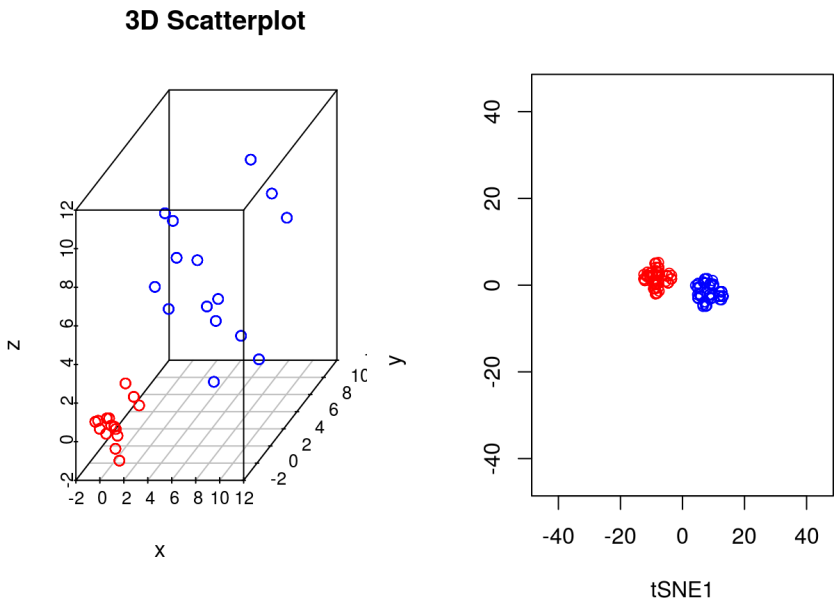

Figura 1.4: Esempio di *Dimensionality Reduction*.

1.2.4 Reinforcement Learning

Il *Reinforcement Learning* è un tipo di algoritmo da domare in modo totalmente diverso.
Un *agente* viene messo all'interno di un ambiente e, in base alle *azioni* che compie, gli verrà assegnata una *ricompensa*, positiva se permette all'*agente* di avvicinarsi all'obiettivo, o negativa nel caso in cui lo allontana. Questo tipo di *Machine Learning* è sensibilmente diverso dagli altri e non ha nulla a che fare con i precedenti.

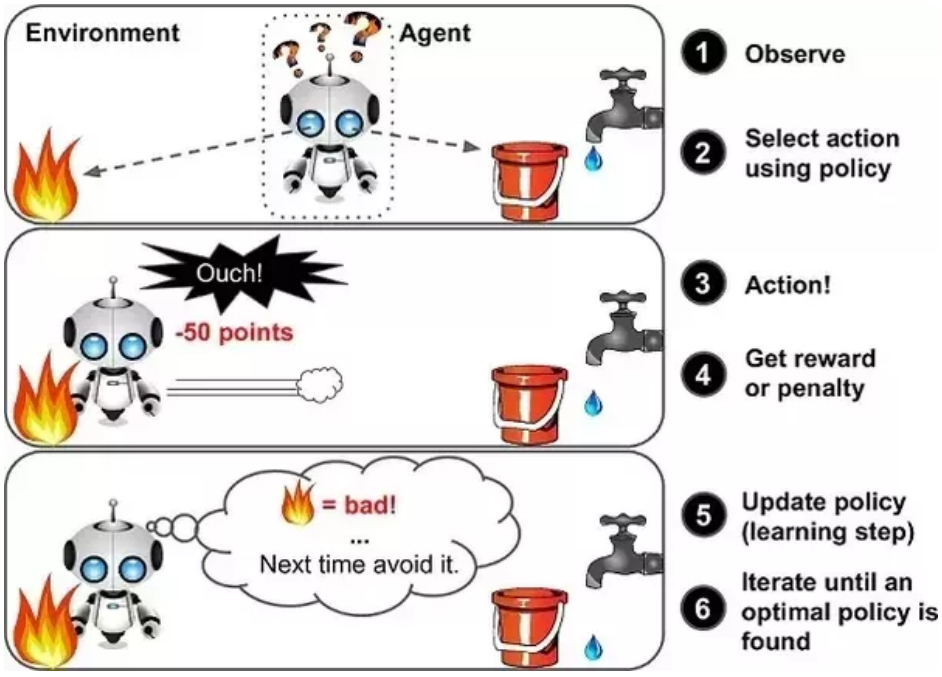

Figura 1.5: Esempio del funzionamento del sistema di rafforzamento.

1.3 Quando affidarsi al *Machine Learning*?

Questa disciplina rappresenta un approccio totalmente differente alla programmazione classica, dove è richiesta grande abilità nel prevedere tutte le possibili situazioni o, come le chiameremo in questo testo, regole. Il *Machine Learning* ha la capacità di adattarsi alle situazioni e ai cambiamenti poichè pone le sua fondamenta sui dati che gli vengono forniti. Di conseguenza, nel caso in cui delle email spam dovessero cambiare forma utilizzando parole differenti, il nostro algoritmo sarebbe in grado di adattarsi e imparare.

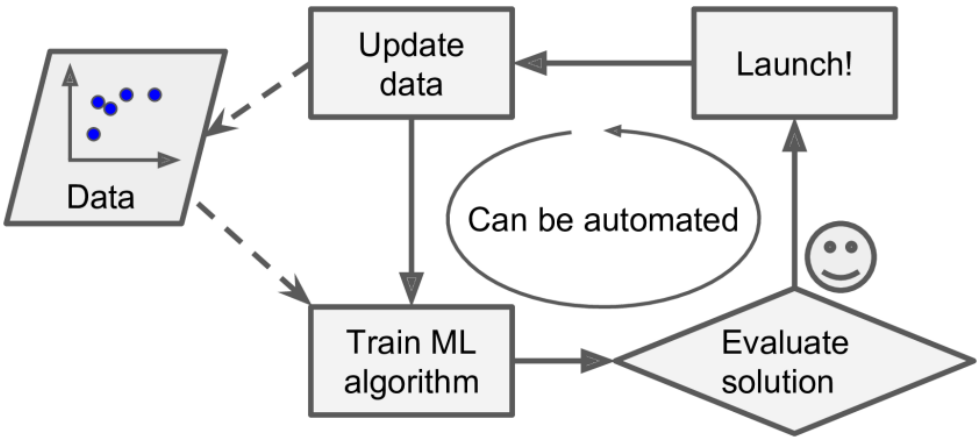

Figura 1.6: Esempio dell'adattamento del *Machine Learning*.

Semplificando, potremmo dire che il *Machine Learning* dovrebbe essere usato per:

- Per problemi che troverebbero soluzione solo attraverso un grandissimo numero di regole, impossibili da prevedere persino per un programmatore esperto.
- Per problemi complessi di cui non si trova una soluzione.
- Per sistemi soggetti a mutare.
- Per ottenere descrizioni dettagliate su enormi quantità di dati.

Capitolo 2

Perceptron

2.1 La storia

Cercando di comprendere il cervello per sviluppare un'intelligenza artificiale, Warren McCullock e Walter Pitts hanno pubblicato nel 1943 uno schema di cellula cerebrale semplificata, il famoso neurone *McCullock-Pitts* (W. S. McCulloch e W. Pitts, *A Logical Calculus of the Ideas Immanent in Nervous Activity. The bulletin of mathematical biophysics*, 5(4):115–133, 1943).
Come sappiamo al giorno d'oggi, i *neuroni* sono cellule nervose interconnesse tra loro che si trovano nel cervello, coinvolte nella trasmissione, ricezione ed elaborazione di segnali elettrici.

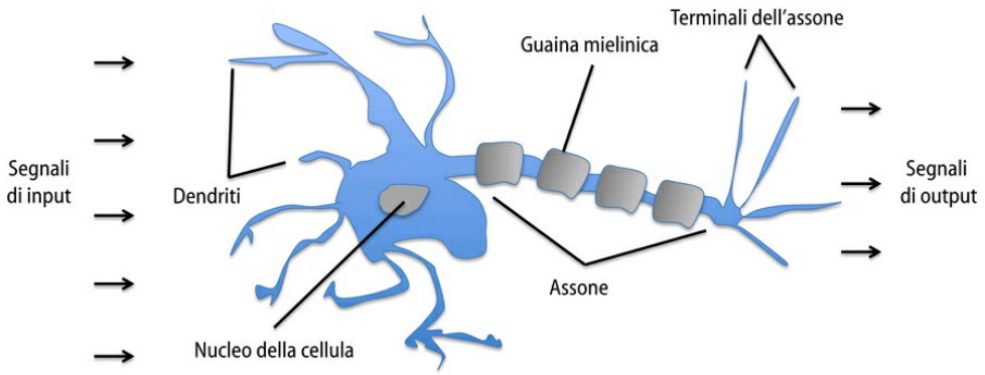

Figura 2.1: Struttura del *neurone*.

I due hanno pensato il neurone come un semplice gate logico con output binario: ai dendriti arrivano più segnali, i quali entrano poi nel corpo della cellula; se il segnale accumulato supera una determinata soglia, viene prodotto un segnale di output, che viene poi passato tramite l'assone.
Pochi anni dopo, Frank Rosenblatt, famoso psicologo, pubblicò il primo concetto di apprendimento automatico sul *Perceptron* basato sul neurone *McCullock-Pitts* (F. Rosenblatt, *The Perceptron, a Perceiving and Recognizing Automaton*, Cornell Aeronautical Laboratory, 1957).
Con la sua regola di apprendimento perceptron, Rosenblatt ha proposto un algoritmo che avrebbe appreso automaticamente i coefficienti di peso ottimali da moltiplicare con le caratteristiche di input in modo da poter prendere la decisione sul fatto che un neurone si attivi o meno.
Nel contesto dell'apprendimento e della classificazione con supervisione, l'algoritmo potrebbe essere utilizzato per prevedere se un determinato campione appartiene a una classe o a un'altra.

2.2 In termini formali

Consideriamo un compito di classificazione binaria, dove la prima classe è etichettata come 1, e l'altra classe è etichettata come -1.
Possiamo definire una funzione di attivazione $\phi(\mathbf{z})$, che prende come valore $\mathbf{z}$, detto *input della rete*, la combinazione lineare di diversi valori input x e il corrispondente vettore di pesi w.

$$\mathbf{x} = \begin{vmatrix} x_0 \\ \vdots \\ x_m \end{vmatrix}, \ \mathbf{w} = \begin{vmatrix} w_0 \\ \vdots \\ w_m \end{vmatrix}$$

Di conseguenza, possiamo definire:

$$z = w_0 x_0 + \ldots + w_m x_m = \sum_{n=0}^{m} w_n x_n = w^T x \tag{2.1}$$

Detto questo, se il valore z supera una certa soglia θ, siamo in grado di prevedere se il campione rientra nella classe 1 o nella classe -1.

Nel *Perceptron*, la funzione di attivazione $\phi(\cdot)$ è una funzione definita a tratti, chiamata solitamente *funzione di passo Heaviside*:

$$\phi(z) = \begin{cases} -1 \text{ if } z < 0 \\ 1 \text{ if } z \geq 0 \end{cases}$$

A sinistra della figura 2.2 troviamo la *Heaviside*, che riduce l'uscita ad un output binario (1,-1); a destra troviamo una possibile separazione delle classi.

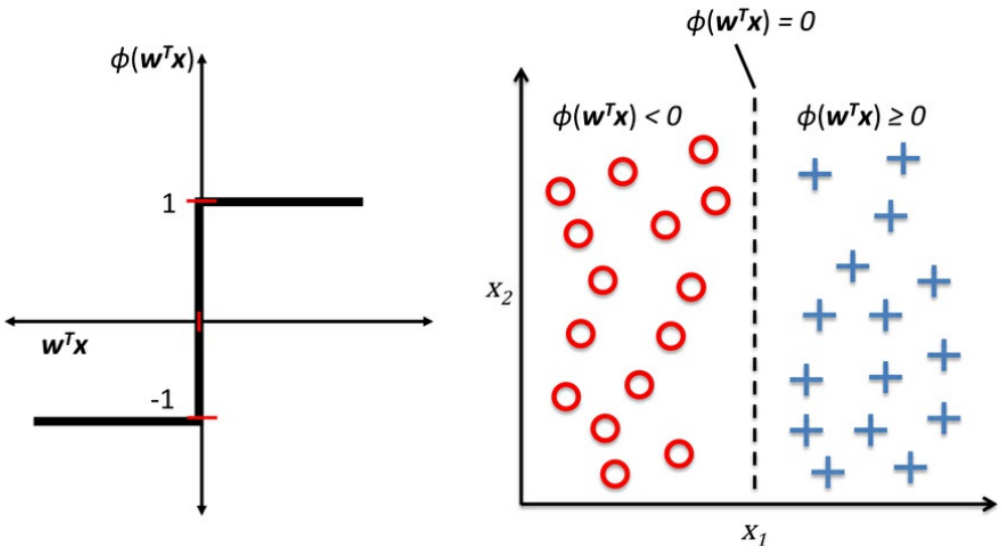

Figura 2.2: Funzione *Heaviside* e separazione lineare delle classi.

Rosenblatt si ispirò alle conoscenze del tempo e, di fatto, un neurone può attivarsi solo nel caso in cui dovesse arrivargli uno stimolo tanto forte da superare la soglia d'attivazione, in caso contrario rimarrebbe spento.
Questo algoritmo non è complesso e potremmo riassumerne i passi in:

1. Inizializzare gli elementi del vettore dei pesi w con valori casuali.
2. Per ogni campione di addestramento x_i svolgere i seguenti passi:
    1. Calcolare il valore di output $\hat{y}$.
    2. Aggiornare i pesi.

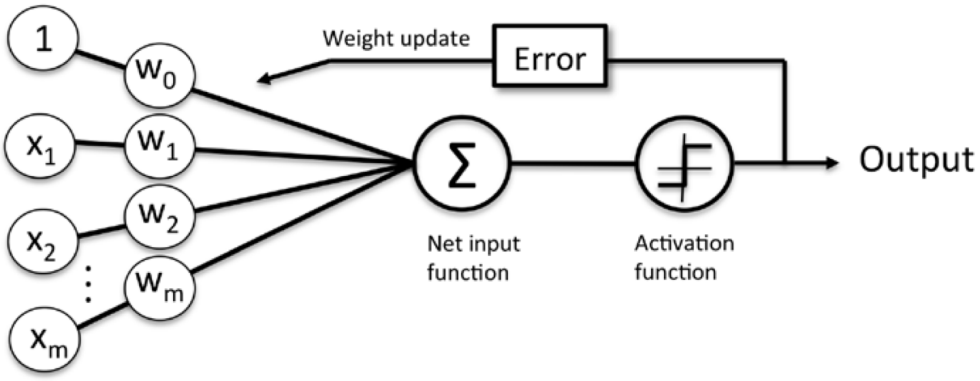

Figura 2.3: Algoritmo del *Perceptron*.

Definiamo l'output $\hat{y}$ come il valore di $\phi(z)$, cioè la funzione definita a tratti *Heaviside*.
Il sopracitato aggiornamento del peso è descritto dalla seguente equazione:

$$w_j = w_j + \Delta w_j \tag{2.2}$$

Abbiamo definito il nuovo peso w_j come il suo valore sommato ad un altro valore scalare, che andremo a calcolare con:

$$\Delta w_j = \eta(y - \hat{y})x_j \tag{2.3}$$

Dove:

- η , chiamato *Learning Rate*, è un valore costante deciso da noi, di solito compreso tra 0.001 e 0.5, utile a definire l'ordine di grandezza di Δw_j. Esso definisce quindi di quanto il peso w_j andrà a modificarsi ogni volta. Se η è un valore troppo alto, i pesi, ad ogni iterazione, andranno a modificarsi di valori troppo grandi, non permettendo all'algoritmo di trovare i valori giusti. Al contrario, se il valore di η è troppo piccolo, ci vorranno troppe iterazioni per raggiungere i valori giusti di w_j;

- y è il valore previsto per quel determinato campione x_i. Questo è il valore in più che ci offre il *Supervised Learning*, ovvero la *label* del campione.

- $\hat{y}$ è l'output calcolato e può assumere due valori (-1, 1);

x_j è il valore dell'input dell'algoritmo;

Notiamo subito che se $\hat{y} = y \Rightarrow \Delta w_j = 0$, quindi non avremo aggiornamento del peso perchè l'algoritmo ha avuto un output uguale a quello previsto, di conseguenza ha predetto correttamente il valore.
Questo procedimento continua fino a quando non viene trovata una configurazione di pesi che riesca a predire correttamente ogni tipo di situazione, purché essa sia linearmente separabile. Ciò che viene tracciata è una retta in grado di separare perfettamente sul piano cartesiano le due classi perciò, nel caso in cui dovessimo avere delle classi "mescolate", non riusciremmo a trovare una retta in grado di separle, e questo algoritmo risulterebbe inutile.

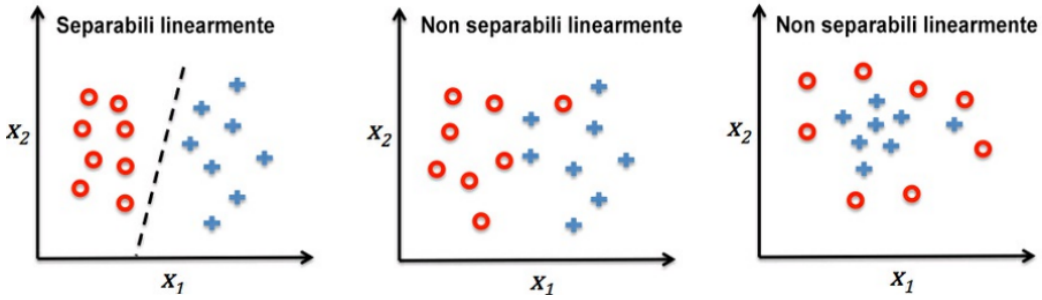

Figura 2.4: DIfferenza tra classi separabili linearmente e non separabili linearmente.

Le iterazioni, nel caso delle classi separabili linearmente, possono continuare fino a quando la condizione di $E_t \neq 0$ è rispettata, ovvero se l'errore totale calcolato su tutti i campioni è diverso da 0 allora continuerà ad aggiustare i pesi, altrimenti avrà trovato la configurazione esatta.
Nel caso in cui non dovessero essere linearmente separabili, si potrebberò definire un numero di iterazioni massime, dette *epoche* o *epochs*, altrimenti l'algoritmo continuerebbe ad aggiustare i pesi all'infinito.