Corso apprendimento automatico

Antonio Montano

2024-03-30

Table of contents

Prefazione

Part I

Introduzione

Part II

1

1 Apprendimento automatico

Un'introduzione

2 1

Part III

Basi dell'apprendimento supervisionato

3 Percettrone

Un'introduzione

3.1 Obiettivo della lezione

1. Studiamo un algoritmo che è una pietra miliare dell'apprendimento delle macchine e che presenta concetti come l'addestramento supervisionato, l'ingegneria delle caratteristiche e la generalizzazione, che sono del tutto generali.
2. È semplice abbastanza perché si possa essere introdotti in uno dei problemi più importanti dell'apprendimento delle macchine, cioè quello della classificazione lineare, nel suo caso più semplice, quella binaria.

3.2 Prerequisiti

1. Conoscenza basilari di geometria, algebra e teoria delle funzioni di più variabili.
2. Capacità di comprensione delle componenti principali di un algoritmo.

3.3 Introduzione

Il `percettrone`, introdotto pubblicamente per la prima volta da Frank Rosenblatt nel 1958, ha giocato un ruolo cruciale nello sviluppo dell'apprendimento delle macchine. Nonostante sia uno dei modelli più semplici di apprendimento supervisionato, la sua introduzione ha segnato l'inizio di un'epoca di grande interesse e sviluppo nel campo delle reti neurali e dell'apprendimento automatico.
Il percettrone è, in breve, un algoritmo per l'`apprendimento supervisionato di classificazioni binarie`. In altre parole, riceve in ingresso un vettore di caratteristiche di un modello fisico e produce un singolo risultato binario. Questo risultato è determinato dalla somma pesata degli ingressi, che passa attraverso una funzione opportuna, detta di attivazione per riferimento al neurone di McCulloch-Pitts, che ha la forma a gradino

per restituire un risultato binario.
La forza del percettrone risiede nella sua capacità di apprendere i pesi ottimali dai dati di addestramento, cioè dati di cui è noto perfettamente sia il vettore di ingresso che il risultato (cioè la classe binaria, dato che tali risultati possono essere solo due). L'algoritmo di apprendimento, per ogni campione di addestramento, predice la classe di risultato. Se la predizione è corretta, i pesi non vengono modificati. Se la predizione è errata, i pesi vengono aggiornati aggiungendo o sottraendo il vettore di ingresso, a seconda che la predizione sia stata troppo bassa o troppo alta.
Rosenblatt ha dimostrato che se i dati di addestramento sono linearmente separabili, allora l'algoritmo di addestramento convergerà a una soluzione che classifica correttamente tutti i campioni di addestramento, in un tempo finito. Block e Novikov hanno anche determinato un limite superiore a tale tempo. Nonostante le sue limitazioni, come l'incapacità di gestire dati che non sono linearmente separabili, l'importanza del percettrone non deve essere sottovalutata. Ha introdotto l'idea fondamentale che le macchine possono apprendere da dati, e ha gettato le basi per lo sviluppo di modelli di apprendimento automatico più sofisticati, tra cui le reti neurali multistrato e le macchine a vettori di supporto.
Infatti, prima del percettrone, i modelli di neuroni artificiali, come il modello di McCulloch-Pitts, erano statici, nel senso che i loro pesi (o le loro connessioni sinaptiche) erano fissi e non cambiavano nel tempo. Questi modelli potevano eseguire calcoli, ma non erano in grado di adattarsi o apprendere dai dati, cioè la conoscenza era 'predefinita' nella rete neurale. Rosenblatt ha introdotto l'idea che i pesi delle connessioni sinaptiche possano essere modificati in base all'esperienza, in modo simile a come si pensava che i neuroni biologici si adattassero e apprendessero nel cervello. Questo è stato un passo fondamentale verso la creazione di modelli di apprendimento delle macchine che potessero imparare dai dati e migliorare così le loro prestazioni nel tempo.
Rosenblatt è probabile che abbia chiamato questo algoritmo, `perceptron` per sottolineare la sua capacità di 'percepire' la struttura nei dati di ingresso. Infatti, la parola perceptron richiama 'perception', percezione, che è il processo cognitivo utilizzato per interpretare le informazioni sensoriali.

The Shallow and the Deep –> LIBRO

3.4 Descrizione dell'algoritmo

Innanzitutto, distinguiamo tre fasi di esecuzione dell'algoritmo: 1. `Fase di addestramento`: usiamo un certo numero di coppie di ingressi e corrispondente uscita, che chiameremo, rispettivamente, caratteristiche e classe, per calcolare dei parametri dell'algoritmo, i pesi, fino a che una certa condizione di qualità globale dell'algoritmo non sia soddisfatta. 2. `Fase di validazione`: adoperando delle coppie non utilizzate in addestramento, si valuta la qualità della predizione senza modificare i parametri del'algoritmo. 3. `Fase di generalizzazione`: diamo in ingresso all'algoritmo delle caratteristiche di cui non conosciamo a priori la classe 'vera' e ne otteniamo una 'predetta'.

In generale, le caratteristiche presenti in ogni ingresso sono in un numero prefissato positivo, che può essere anche arbitrariamente grande, mentre l'uscita sarà sempre un valore unico scelto tra due possibilità. Se rappresentiamo ogni ingresso con $\mathbf{x}$, la corrispondente classe 'vera' con y, i pesi con $\mathbf{w}$ e, infine, la classe predetta con $\bar{y}$, allora nel diagramma seguente (Figura 1) è riassunta la fase di addestramento del percettrone. {{ insert_image(target, '1', image_location, 'Percettrone-Diagramma-addestramento.png', 'Figura 1: Diagramma del percettrone in fase di addestramento', 'Diagramma del percettrone in fase di addestramento') }}

In ogni esecuzione di addestramento, forniremo in ingresso un gruppo di caratteristiche distinte prese dalla totalità degli ingressi disponibili, di cui conosciamo anche le rispettive classi. Tutte le coppie di caratteristiche e relative classi compongono l'**`insieme di addestramento`** e ogni suo elemento si chiama **`campione`**. Nel diagramma le caratteristiche entrano in un nodo dove viene calcolato il valore z, ottenuto come somma pesata delle caratteristiche e usando dei pesi ricavati dall'esecuzione appena precedente. z è, a sua volta, l'ingresso di un secondo nodo con la funzione $\mathcal{H}$, detta `funzione di attivazione`, che ne assegna la classe corrispondente.
Nel caso del percettrone, la funzione di attivazione è tale che per valori non negativi la classe è $+1$, altrimenti, per quelli negativi, è -1. La funzione $\mathcal{H}$ restituisce sempre valori numerici, ma, generalmente, ad ognuno è associato, in esclusiva, un oggetto del modello fisico che ha un nome distintivo, definito come `etichetta`. Classe ed etichetta sono usati in modo intercambiabile, data la corrispondenza 1 ad 1. Il valore predetto $\bar{y}$ viene confrontato in un terzo nodo col valore reale y corrispondente al vettore di caratteristiche $\mathbf{x}$: se i due valori concidono, quindi la predizione è esatta, allora i pesi non vengono aggiornati, sennò lo sono secondo una semplice formula che prevede di sommare al valore della precedente esecuzione, il prodotto tra un valore costante $\pm 2\eta$, positivo se il valore corretto era $+1$, negativo in caso contrario, con $\mathbf{x}$. Ogni esecuzione è una `iterazione` di un ciclo che prende in ingresso un nuovo campione e il valore dei pesi testé calcolato e continua fino a che non diventi vera una `condizione di terminazione`.
Ma quando terminano le iterazioni? L'addestramento terminerà quando la somma degli errori su tutti i campioni sarà nullo, oppure al disotto di un valore prefissato. Nel mentre, i campioni dell'insieme di addestramento saranno usati tutti in un gruppo di iterazioni, definito come `epoca`, il cui numero è, evidentemente, pari al numero di campioni dell'insieme di addestramento.
Al termine dell'addestramento sarà ottenuto un insieme di pesi, la cui numerosità è pari a quella delle caratteristiche, e tali valori saranno usati, senza alcuna ulteriore modifica, in tutte le esecuzioni della seguente fase di generalizzazione (Figura 2), in cui useremo il percettrone per ottenere classi a priori ignote. {{ insert_image(target, '2', image_location, 'Percettrone-Diagramma-generalizzazione.png', 'Figura 2: Diagramma del percettrone in fase di generalizzazione', 'Diagramma del percettrone in fase di generalizzazione') }}

3.5 Funzione di attivazione

Innanzitutto, definiamo formalmente la funzione di attivazione a gradino $\mathcal{H}$ (che è una versione modificata della `funzione di Heaviside`):

$$\mathcal{H}(z) = \begin{cases} 1, & z \geq \theta \\ -1, & z < \theta \end{cases} \tag{1}$$

il cui grafico è mostrato in Figura 3. Il significato è il seguente: se il valore dell'argomento della funzione è maggiore o uguale a θ, allora essa assumerà il valore di $+1$. Se invece l'argomento sarà minore di θ, il risultato sarà -1. {{ insert_image(target, '3', image_location, 'Percettrone-Funzione-attivazione-gradino.png', 'Figura 3: Funzione di attivazione a gradino', 'Funzione di attivazione a gradino') }}

Nella definizione è presente una soglia predefinita θ, che permette di inserire un grado di libertà molto importante nell'addestramento, di cui approfondiremo nel seguito. L'argomento della funzione di attivazione è $z = \sum_{j=1}^{n} w_j x_j$, ottenuto come somma pesata delle caratteristiche in ingresso alla iterazione, ove n è il numero delle caratteristiche.
Generalmente, si preferisce una formulazione del tutto equivalente in cui $\mathcal{H}$ ha il gradino nello 0 dell'asse delle ascisse. A tal fine, definiamo $\bar{z} = z - \theta$ e riformuliamo la funzione di attivazione:

$$\mathcal{H}(\bar{z}) = \begin{cases} 1, & \bar{z} \geq 0 \\ -1, & \bar{z} < 0 \end{cases}, \ \bar{z} = \sum_{j=1}^{N} w_j x_j - \theta \tag{2}$$

e definendo $w_0 = -\theta$ (chiamato **`deriva`**) e $x_0 = 1$, quindi rinominando $\bar{z}$ in z, si ricava:

$$\mathcal{H}(z) = \begin{cases} 1, & z \geq 0 \\ -1, & z < 0 \end{cases}, \ z = \sum_{j=0}^{N} w_j x_j \tag{3}$$

dove, sostanzialmente, abbiamo esteso la sommatoria per comprendere anche θ.
Questa definizione può essere resa in formato più compatto, introducendo i vettori $\mathbf{w}$ per i pesi e $\mathbf{x}$ per le caratteristiche:

$$\mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix}, \ \mathbf{x} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{pmatrix} \tag{4}$$

dove ricordiamo che le caratteristiche 'reali' del modello sono sempre in numero di n, mentre i due vettori hanno dimensione $n + 1$. Pertanto, la funzione di attivazione può essere riscritta utilizzando il prodotto scalare tra i due vettori dei pesi e delle caratteristiche, cioè $z = \sum_{j=0}^{N} w_j x_j = \langle \mathbf{w}, \mathbf{x} \rangle$:

$$\mathcal{H}(z) = \begin{cases} 1, & z \geq 0 \\ -1, & z < 0 \end{cases}, \ z = \langle \mathbf{w}, \mathbf{x} \rangle. \tag{5}$$

3.6 Regola di apprendimento

Nella fase di addestramento, il percettrone 'apprende' i pesi che saranno usati in generalizzazione, cioè l'algoritmo determina il vettore $\mathbf{w}$, sotto la `supervisione` di un 'insegnante' che fornisce l'insieme di addestramento e osserva le predizioni (donde la caratteristica dell'algoritmo di `apprendimento supervisionato`).
Introduciamo alcune notazioni: l'insieme di addestramento Ξ sia composto da N elementi $(\mathbf{x}^i, y_i)$ (i campioni), ognuno con due valori, il vettore delle $n+1$ caratteristiche reali $\mathbf{x}^i$ e la sua classe y^i che può assumere solo i valori ± 1:

$$(\mathbf{x}^1, y_1), \ldots, (\mathbf{x}^N, y_N) \in \Xi \tag{6}$$

con N numero positivo, η sia il **`tasso di apprendimento`** con $0 < \eta \leq 1$, K sia un numero intero positivo che identifichi l'iterazione corrente, e $\bar{y}_K$ denoti il risultato dell'algoritmo all'iterazione K, cioè la classe predetta, allora i passi che il percettrone esegue nella fase di apprendimento, sono riassunti nella cosiddetta **`regola di apprendimento`**:

> **`Regola di apprendimento del percettrone`** 1. Inizializza i pesi $\mathbf{w}^0$ a 0 oppure a piccoli valori casuali, definisci un tasso di apprendimento η e un numero massimo di iterazioni $\bar{K}$ o un numero massimo di epoche $\bar{E}$. 2. Per ogni epoca E: 1. Per ogni campione $(\mathbf{x}^\kappa, y_\kappa)$ dell'insieme di addestramento ($1 \leq \kappa \leq N$ è l'indice positivo sull'insieme di addestramento, E è l'indice positivo di epoche, N la dimensione dell'insieme di addestramento e $K = EN + \kappa$ il totale delle iterazioni compiute): 1. Calcola la predizione della classe corrispondente a $\mathbf{x}^\kappa$:
>
> $$\bar{y}_K = \mathcal{H}\left(\sum_{j=0}^{n} w_j^{K-1} x_j^\kappa\right). \tag{7}$$
>
> 2. Aggiorna i pesi $\mathbf{w}^K$ secondo la formula:
>
> $$\mathbf{w}^K = \mathbf{w}^{K-1} + \eta(y_\kappa - \bar{y}_K)\mathbf{x}^\kappa. \tag{8}$$
>
> 2. Continua fino alla prima occorrenza di una tra le due condizioni seguenti: non vi siano più errori di predizione sull'insieme di addestramento, cioè $y_\kappa - \bar{y}_K = 0\ \forall\kappa$, cioè in una intera epoca, oppure finché non sia raggiunto il numero massimo di iterazioni $\bar{K}$ o il numero massimo di epoche $\bar{E}$ (dove $\bar{K} = N\bar{E}$). 3. Prendi il vettore dei pesi $\mathbf{w}$ per classificare caratteristiche con etichetta ignota nella fase di generalizzazione.

Il tasso di apprendimento η e il numero massimo di iterazioni $\bar{K}$ sono detti **`iperparametri`**, dato che non sono modificati dalla regola di apprendimento, ma ricavati con tecniche aggiuntive ad hoc. Le iterazioni sono tali da applicare tutto l'insieme di addestramento un certo numero intero di volte, quindi $\bar{K} = E \cdot N$.

Si nota che: * Se il percettrone predice correttamente la classe, allora $y_\kappa - \bar{y}_K = 0$, quindi i pesi rimangono invariati. * Se il percettrone effettua una predizione erronea, allora

$$\mathbf{w}^K - \mathbf{w}^{K-1} = \eta(y_\kappa - \bar{y}_K)\mathbf{x}^\kappa = \begin{cases} \eta(1-(-1))\mathbf{x}^\kappa = 2\eta\mathbf{x}^\kappa & y_\kappa = 1,\ \bar{y}_K = -1 \\ \eta(-1-(1))\mathbf{x}^\kappa = -2\eta\mathbf{x}^\kappa & y_\kappa = -1,\ \bar{y}_K = 1 \end{cases} \quad 9$$

quindi per $y_\kappa = 1$ i pesi si incrementano di una frazione positiva del vettore delle caratteristiche, mentre per $y_\kappa = -1$ di una frazione negativa, la cui entità dipende da η.

3.7 Interpretazione geometrica

Possiamo interpretare la 5, come l'esistenza di una frontiera, per cui se un vettore delle caratteristiche è 'sopra' di essa, allora la classe corrispondente è $+1$, se al disotto allora sarà -1. Tale frontiera è definita da $\langle \mathbf{w}, \mathbf{x} \rangle = 0$ o, equivalentemente $\sum_{j=0}^{n} w_j x_j = 0$, che possiamo riscrivere, ricordando che $x_0 = 1$, come $w_0 + x_1 w_1 + \cdots + x_n w_n = 0$ che corrisponde ad un punto su una retta per $n = 1$, una retta nel piano per $n = 2$, un piano nello spazio tridimensionale per $n = 3$ e un iperpiano in $\mathbb{R}^n$ per $n > 3$.
Scegliamo $n = 2$ per poter visualizzare i vettori delle caratteristiche e la frontiera. Un vettore $\mathbf{x}$, così come definito in 4, lo disegniamo come un punto di coordinate (x_1, x_2) e la frontiera di equazione $w_0 + x_1 w_1 + x_2 w_2 = 0$, come una retta. Sappiamo che, ad ogni iterazione, i tre pesi w_0, w_1, w_2 possono assumere dei nuovi valori e, al termine dell'addestramento, avranno quelli definitivi da usare nella fase di generalizzazione.
Nella Figura 4, tutti i punti sopra o sulla retta (sfondo rosa) sono tali che $w_0 + x_1 w_1 + x_2 w_2 \geq 0$, quindi hanno classe $+1$, mentre per quelli per cui è $w_0 + x_1 w_1 + x_2 w_2 < 0$ la classe è -1 (sfondo azzurro). {{ insert_image(target, '4', image_location, 'Percettrone-Frontiera-decisione.png', 'Figura 4: Spazio bidimensionale delle caratteristiche e frontiera di decisione', 'Spazio bidimensionale delle caratteristiche e frontiera di decisione') }}

Quindi, la retta $w_0 + x_1 w_1 + x_2 w_2 = 0$ si comporta come una vera e propria **`frontiera di decisione`**, giacché la posizione relativa del vettore delle caratteristiche rispetto alla retta, determina la classe predetta che gli corrisponde. Proprio perché la frontiera di decisione è una linea, allora il percettrone è definito come un **`classificatore lineare`**.
In pratica, l'addestramento consiste nel muovere la retta (o per un numero di caratteristiche $n > 2$, un piano o un iperpiano) nello spazio bidimensionale (rispettivamente, nello spazio tridimensionale o multidimensionale), in modo da dividerlo perché tutti i punti con classe $+1$ siano sopra la frontiera e i rimanenti al disotto. Ciò non è detto sia a priori possibile, a causa della distribuzione delle caratteristiche, cioè non esista tale frontiera come nel caso della Figura 5. {{ insert_image(target, '5', image_location, 'Percettrone-Caratteristiche-non-lin-sep.png', 'Figura 5: Caratteristiche non linearmente separabili', 'Caratteristiche non linearmente separabili') }}

Ciò si esprime dicendo che l'insieme di addestramento non è **`linearmente separabile`** o, in altre parole, che il percettrone commetterà sempre degli errori di classificazione in fase di

addestramento, giacché, per costruzione, può solo produrre frontiere di decisione lineari. Al contrario, in Figura 4, è mostrato un insieme di addestramento linearmente separabile con una possibile, tra le infinite, frontiera di separazione che efficacemente distingue le caratteristiche (in Figura 6 alcuni esempi di frontiere di decisione alternative). {{ insert_image(target, '6', image_location, 'Percettrone-Frontiere-multiple.png', 'Figura 6: Frontiere di decisione multiple', 'Frontiere di decisione multiple') }}

3.8 Deriva

Il termine deriva è usato in matematica e in informatica per rappresentare una sorta di correzione, che viene aggiunta al risultato di un algoritmo. Nella regola di apprendimento del percettrone, si nota che senza la deriva w_0, la frontiera di decisione passerebbe sempre per l'origine dello spazio delle caratteristiche e ciò ne limiterebbe evidentemente la capacità di separazione di sottoinsiemi degli insiemi di apprendimento.
Per visualizzare questa affermazione, poniamo sempre $n = 2$ e facendo riferimento alla Figura 6, dove è disegnata la frontiera $w_0 + x_1 w_1 + x_2 w_2 = 0$ con la sua intersezione con l'asse delle ordinate $-\frac{w_0}{w_2}$ che si può esprimere anche con $\frac{\theta}{w_2}$, ci si può convincere che per $w_0 = 0$ o $\theta = 0$, la frontiera passa per l'origine delle coordinate dello spazio delle caratteristiche. {{ insert_image(target, '7', image_location, 'Percettrone-Frontiera-decisione-intercetta.png', 'Figura 7: Frontiera di decisione e deriva', 'Frontiera di decisione e deriva') }}

3.9 Convergenza

Sappiamo che una condizione necessaria perché il percettrone possa classificare correttamente l'insieme di addestramento, sia che questo abbia la proprietà di lineare separabilità. Quello che non sappiamo è se ciò sia anche sufficiente, cioè se è garantito che la regola di apprendimento produca un vettore **w** senza errori di classificazione e, quindi, sia costruita una frontiera di separazione efficace. E questa è proprio la tesi del teorema di convergenza del percettrone pubblicato per la prima volta da Rosenblatt nel 1958 ([RF58II] e [RF62]). > **`Teorema di convergenza del percettrone`**
> > Per ogni insieme di addestramento, composto da un numero finito di campioni, l'algoritmo di addestramento del percettrone produrrà un vettore **w** che classificherà correttamente tutti tali elementi, in un numero finito di epoche.

Ciò si esprime anche dicendo che l'algoritmo **`converge`** e, usualmente, ciò accade in un numero di iterazioni che è di gran lunga superiore alla dimensione dell'insieme di addestramento. Se la condizione del teorema non è soddisfatta, allora l'algoritmo continuerà a fare aggiustamenti ai pesi e alla deriva, senza mai raggiungere una soluzione che classifichi correttamente tutti i campioni dell'insieme di addestramento. In questo caso, si dice che l'algoritmo non converge.
Un secondo teorema, più tecnico, dimostrato indipendentemente da Henry Block [BH62] e

Aleksey Novikov [NA62], stabilisce un limite superiore al numero di errori di classificazione, il che ci permette di stimare il tempo necessario all'algoritmo per convergere nel caso di insieme di addestramento linearmente separabile.

> **`Teorema del numero massimo di errori del percettrone (Block/Novikov)`**
>
> Assumiamo che l'insieme di addestramento Ξ sia linearmente separabile con margine γ e che la lunghezza (o norma euclidea) di tutti i vettori delle caratteristiche, sia minore o uguale di un certo valore R finito (o, più formalmente, $\|\mathbf{x}^i\| \leq R \ \forall i$, per $R > 0$). Quindi, il massimo numero di errori compiuti dall'algorimo del percettrone in fase di addestramento, è limitato superiormente da $\frac{R^2\|\mathbf{w}\|^2}{\gamma^2\|\widetilde{\mathbf{w}}\|^2}$, con $\mathbf{w}$ vettore dei pesi a convergenza e $\widetilde{\mathbf{w}}$ corrispondente vettore normale alla frontiera di separazione.

Nel teorema si cita il **`margine`** che è definito come la distanza minima delle caratteristiche dalla frontiera di decisione in $\mathbb{R}^n$.
Quindi, adesso sappiamo che il numero massimo di epoche di addestramento è limitato superiormente e se si commettesse almeno un errore per epoca (se gli errori fossero zero allora l'algoritmo avrebbe raggiunto la convergenza), allora il numero di epoche sarebbe inferiore a $\frac{R^2\|\mathbf{w}\|^2}{\gamma^2\|\widetilde{\mathbf{w}}\|^2}$. Ciò è senz'altro confortante ma potrebbe portare ad un valore molto alto e, d'altronde, l'utilità maggiore del teorema di Block-Novikov è nel notare che il limite al numero massimo di errori che esso definisce, è indipendente da: * Numero di dimensioni dello spazio delle caratteristiche. * Dimensione dell'insieme di addestramento, cioè dal numero dei campioni. * Tasso di apprendimento fintantoché sia mantenuto costante (meno ovvio perché non citato nella tesi, ma ipotesi nella dimostrazione). * Valore del vettore di inizializzazione di $\mathbf{w}$. * Ordine dei campioni dell'insieme di addestramento.

Al contrario, è dipendente da: * 'Difficoltà' nella separazione, cioè avere γ molto bassi (rispetto ai valori delle caratteristiche) incrementa il limite. * La dispersione dei campioni (che porta ad un R elevato).

Il teorema di Block-Novikov è interessante perché pone l'accento sull'importanza di alcune proprietà dell'insieme di addestramento rispetto ad altre, ma è, al contempo, poco utile operativamente. Infatti, per stimare il limite superiore agli errori abbiamo bisogno di fornire il margine η, che non è noto a priori e non è un risultato dell'addestramento!
D'altronde, anche ottenendo una stima della frontiera di decisione e del margine, comunque non avremmo alcuna garanzia che siano 'vicine' ai valori di convergenza ottenuti dall'addestramento dell'algoritmo e, soprattutto, né che tali valori siano ottimali, cioè tali da avere il margine massimo o massimizzare una qualsiasi altra metrica di 'qualità', sia in addestramento che nella fase seguente di generalizzazione. In generale, il percettrone convergerà ad una delle possibili frontiere di decisione che separano i due insiemi di caratterstiche corripondenti a classi diverse (come visualizzato in Figura 6).

3.10 Generalizzazione

Dopo l'addestramento supervisionato, il percettrone è pronto per classificare ingressi di cui la classe è ignota. La fase diventa di **`generalizzazione`**, definita come la capacità di estendere la conoscenza del modello acquisita per mezzo dei campioni dell'insieme di addestramento, a nuovi ingressi di cui le relative classi sono sconosciute, coll'obiettivo di predire proprio quest'ultime, in modo corretto. Questa finalità è la ragione per cui il percettrone e tutti gli altri algoritmi di classificazione, sono sviluppati.
Possiamo definire una regola di generalizzazione, al pari di quella di apprendimento, per definire il calcolo della predizione dato un nuovo ingresso di caratteristiche.

> **`Regola di generalizzazione del percettrone`** Dato un vettore di caratteristiche **`x`**, la predizione sarà pari a:
>
> $$\bar{y} = \mathcal{H}\left(\sum_{j=0}^{n} w_j x_j\right). \tag{10}$$

D'altronde, la capacità del percettrone è limitata dall'essere un classificatore lineare, il che significa che può solo apprendere relazioni lineari tra le caratteristiche, quindi se la loro relazione intrinseca fosse non lineare, esso non sarebbe comunque in grado di apprenderla accuratamente e, quindi, avrebbe una scarsa capacità di generalizzazione, o, in altre parole, commetterebbe un certo numero di errori di predizione. Inoltre, anche la qualità dei dati di addestramento ha un impatto sulla capacità di generalizzazione. Se i campioni contengono errori o rumore, o se non sono rappresentativi dell'intera popolazione, il percettrone non avrà una base informativa sufficiente.
Un altro fattore che influisce sulla capacità di generalizzazione è la complessità del modello fisico, che è legata al numero di caratteristiche. Se il numero di caratteristiche è troppo grande rispetto al numero di campioni, l'algoritmo può finire per `sovradattarsi` ai dati di addestramento, il che significa che apprende perfettamente i dati di addestramento, ma mostra una efficacia limitata o, comunque, nettamente inferiore nella fase di validazione, intermedia tra addestramento e generalizzazione vera e propria. In generale, esistono diverse cause di sovradattamento, che sono meglio connotate con algoritmi più complessi, ma quello che è importante sottolineare è che il percettrone non ne è immune anche se, concettualmente, è un algoritmo molto semplice.
L'errore commesso nella fase di validazione, è denominato **`errore di generalizzazione`** e il suo calcolo si effettua prendendo un insieme di campioni di caratteristiche e relative classi, non usate in fase di addestramento, e valutando le predizioni senza che i pesi e la deriva siano ulteriormente aggiornati. Quello che si ottiene rappresenta una metrica di qualità dell'algoritmo nel suo complesso, cioè, contemporaneamente, della scelta delle caratteristiche del modello e della regola di apprendimento utilizzata, cioè dell'algoritmo.

3.11 Insiemi di addestramento non separabili

Se l'insieme di addestramento non è linearmente separabile allora, per sua definizione, non esiste alcuna frontiera di decisione lineare che possa dividere l'insieme in due regioni dove, in ognuna, siano presenti solo caratteristiche corrispondenti alla medesima classe. In questo caso, il percettrone non potrà convergere e, ad ogni iterazione nell'addestramento, produrrà una frontiera con un certo numero di errori associati, qualsiasi sia il numero di iterazioni eseguite. Questa è una forma di **`sottoadattamento`**, cioè il modello è troppo semplice per catturare la struttura sottostante dei dati e ciò in un modo aprioristico: è l'algoritmo utilizzato che non lo permette.
Già Marvin Minsky e Seymour Papert, in [MP69], avevano mostrato che, in caso di separabilità non lineare, il percettrone ricalcolerà continuamente lo stesso vettore di pesi, quindi entrando in un ciclo infinito. Ciò comporta la necessità di algoritmi più avanzati per classificare correttamente tali insiemi di addestramento, come le **`macchine a vettori di supporto`**, o altri metodi più avanzati del percettrone.

3.12 Ordine dei campioni in fase di addestramento

Durante l'addestramento, il percettrone aggiorna iterativamente i suoi pesi in base agli errori commessi. Questi aggiornamenti sono guidati dalla sequenza di campioni di addestramento forniti. Ecco perché l'ordine in cui questi campioni vengono presentati, può influenzare significativamente la velocità di convergenza e tecniche di modifica di tale ordine riducono significativamente le iterazioni necessarie al raggiungimento della soglia di arresto.
Rispetto alla regola di apprendimento presentata precedentemente, un solo cambiamento viene effettuato e cioè l'aggiunta di un passo nell'iterazione sulle epoche, in cui adottiamo una strategia di modifica, o anche una combinazione, dell'ordine dei campioni, che può includere anche una modifica nel numero di iterazioni nell'epoca, nel caso in cui uno o più campioni vengano omessi o presentati più volte.

> **`Regola di apprendimento del percettrone con ordine dei campioni modificato`** 1. Inizializza i pesi $\mathbf{w}^0$ a 0 oppure a piccoli valori casuali, definisci un tasso di apprendimento η e un numero massimo di iterazioni $\bar{K}$ o un numero massimo di epoche $\bar{E}$. 2. Per ogni epoca E: 1. Modifica l'ordine dei campioni dell'insieme di addestramento 2. Per ogni campione $(\mathbf{x}^\kappa, y_\kappa)$ dell'insieme di addestramento ($1 \leq \kappa \leq N$ è l'indice positivo sull'insieme di addestramento, E è l'indice positivo di epoche, N la dimensione dell'insieme di addestramento e $K = EN + \kappa$ il totale delle iterazioni compiute): 1. Calcola la predizione della classe corrispondente a $\mathbf{x}^\kappa$:
>
> $$\bar{y}_K = \mathcal{H}\left(\sum_{j=0}^{n} w_j^{K-1} x_j^\kappa\right). \tag{11}$$

2. Aggiorna i pesi $\mathbf{w}^K$ secondo la formula:

$$\mathbf{w}^K = \mathbf{w}^{K-1} + \eta(y_\kappa - \bar{y}_K)\mathbf{x}^\kappa. \tag{12}$$

3. Continua fino alla prima occorrenza di una tra le due condizioni seguenti: non vi siano più errori di predizione sull'insieme di addestramento, cioè $y_\kappa - \bar{y}_K = 0\ \forall\kappa$, cioè in una intera epoca, oppure finché non sia raggiunto il numero massimo di iterazioni $\bar{K}$ o il numero massimo di epoche $\bar{E}$ (dove $\bar{K} = N\bar{E}$). 3. Prendi il vettore dei pesi $\mathbf{w}$ per classificare caratteristiche con etichetta ignota nella fase di generalizzazione.

Abbiamo aggiunto il passo 2.1 che a seconda delle strategie adottate verrà esploso nelle istruzioni necessarie, anche complesse. Alcune strategie: * Rimescolamento degli indici: gli indici vengono permutati con un algoritmo come quello di Fisher-Yates che garantisce una permutazione imparziale, cioè ogni possibile permutazione dell'elenco degli indici ha la stessa probabilità di occorrenza. È implementato dalla funzione `random.shuffle` in Python. Vi sono alternative più complesse. * Ciclo: in pratica ad ogni epoca si parte dall'elemento successivo a quello usato nella precedente (gli indici sono ottenuti dalla operazione $i + 1 \bmod n$, se i è l'indice dell'ultima sequenza). * Massima distanza: ad ogni epoca dividiamo in due sottoinsiemi l'insieme di addestramento: quello coi campioni non correttamente classificati e il complementare. Il primo lo ordiniamo per distanza decrescente $\frac{-y_\kappa\langle\mathbf{w}^K,\mathbf{x}^\kappa\rangle}{\|\widetilde{\mathbf{w}}^K\|}$ dalla ultima frontiera di decisione (quindi il primo sarà quello 'più' erroneamente classificato). Il secondo sottoinsieme per distanza crescente $\frac{y_\kappa\langle\mathbf{w}^K,\mathbf{x}^\kappa\rangle}{\|\widetilde{\mathbf{w}}^K\|}$ dalla frontiera di decisione o rimescolati. L'ordine nell'epoca sarà dato dal primo sottoinsieme seguito dal secondo. * Massimo residuo: si ordinano i campioni sulla base dell'errore, in valore assoluto, commesso prima dell'applicazione della funzione di attivazione alla predizione $|\langle\mathbf{w}^K, \mathbf{x}^\kappa\rangle - y^\kappa|$, a partire dal massimo e poi via via in modo decrescente.

Da notare come le prime due strategie siano indipendenti dalle caratteristiche, dato che agiscono sui soli indici, mentre quelle di massima distanza e residuo, dipendano dai campioni. Inoltre, possono essere anche combinate come, ad esempio, la massima distanza per i campioni classificati erroneamente e il rimescolamento per quelli correttamente classificati.

3.13 Oltre la classificazione binaria

Nel caso di problemi con un numero di classi maggiore di due, possiamo usare delle tecniche di riduzione al caso binario, tra cui: * **`Uno contro tutti`**: Addestriamo tanti percettroni quante classi e ognuno sarà associato ad una classe che diventerà il 'caso positivo' (classe $+1$) e tutte le altre corrisponderanno al 'caso negativo' (classe -1). Tutti gli insiemi di addestramento risulteranno **`sbilanciati`**, se l'insieme di partenza avrà un numero di campioni per classe simile. Ciò, in generale, vedremo essere una condizione non ottimale. * **`Uno contro uno`**: per ogni coppia di classi, addestriamo un percettrone e, pertanto, in totale saranno $\binom{\Sigma}{2} = \frac{\Sigma(\Sigma-1)}{2}$,