Capitolo 1

Alla ricerca disperata di modelli

Quando era ancora un bambino, lo scienziato austriaco Konrad Lorenz, incuriosito dai racconti di un libro intitolato *Il meraviglioso viaggio di Nils Holgersson* (le avventure di un ragazzo con le oche selvatiche, a opera della svedese Selma Lagerlöf, vincitrice del Premio Nobel per la letteratura), desiderava diventare egli stesso un'oca selvatica. Non potendo assecondare la propria fantasia, il giovane Lorenz si accontentò di prendersi cura di un anatroccolo di un giorno, regalatogli dal vicino. Per la gioia del bambino, l'anatroccolo incominciò a seguirlo ovunque: aveva subìto un imprinting su di lui. Il termine "imprinting" si riferisce alla capacità di molti animali, compresi gli anatroccoli, di sviluppare un legame stretto con il primo movimento che vedono subito dopo la nascita. Lorenz diventerà un etologo e sarà un pioniere degli studi sul comportamento animale. In particolare proprio sull'imprinting: gli anatroccoli lo seguivano mentre camminava, correva, nuotava e persino quando pagaiava su una canoa. Vinse il Premio Nobel per la Fisiologia e la Medicina nel 1973, insieme ai colleghi etologi Karl von Frisch e Nikolaas Tinbergen. I tre vennero celebrati "per le loro scoperte sull'organizzazione e l'elicitazione dei *modelli* di comportamento individuale e sociale".

Si è scoperto che gli anatroccoli possono subire l'imprinting non solo sulla prima creatura vivente che vedono muoversi, ma anche sugli oggetti inanimati. Essi, per esempio, possono legarsi a una coppia di oggetti in movimento, simili per forma o colore. In particolare, il loro imprinting riguarda il concetto relazionale incarnato dagli oggetti. Così, se alla nascita gli anatroccoli vedono due oggetti in movimento, entrambi di colore rosso, in seguito seguiranno due oggetti del medesimo colore, anche se sono entrambi blu e non rossi, ma non due oggetti di colore diverso l'uno dall'altro. In questo caso, gli anatroccoli imprimono nel loro cervello l'*idea* di somiglianza. Però dimostrano anche la loro capacità di discernere la dissomiglianza. Se i primi og-

getti in movimento visti dagli anatroccoli sono, per esempio, un cubo e un prisma rettangolare, riconosceranno che tali oggetti hanno forme diverse e seguiranno, successivamente, due oggetti di forma diversa (una piramide e un cono, per esempio), ma ignoreranno due oggetti che hanno la stessa forma.

Riflettiamo per un momento. Gli anatroccoli appena nati, con una brevissima esposizione agli stimoli sensoriali, individuano schemi in ciò che vedono, formano nozioni astratte di somiglianza e dissomiglianza e poi riconoscono queste astrazioni negli stimoli che vedono in seguito, agendo di conseguenza. I ricercatori sull'intelligenza artificiale offrirebbero un occhio della testa per sapere come gli anatroccoli riescano a fare tutto questo.

Sebbene l'AI di oggi sia ben lontana dall'essere in grado di svolgere tali compiti con la facilità e l'efficienza degli anatroccoli, possiede qualcosa in comune con questi ultimi, ovvero la capacità di *individuare* e *imparare* i modelli presenti nei dati. Quando Frank Rosenblatt inventò il "perceptron" alla fine degli anni '50, uno dei motivi per cui fece tanto scalpore fu che si trattava del primo formidabile algoritmo "ispirato al cervello", in grado di imparare a riconoscere gli schemi nei dati semplicemente esaminandoli. Soprattutto, date alcune ipotesi sui dati, i ricercatori dimostrarono che il perceptron di Rosenblatt trovava sempre il modello nascosto nei dati in un tempo *finito*. In altre parole, il perceptron convergeva su una soluzione. Queste certezze, in informatica, sono preziose come l'oro. Non c'è da stupirsi che l'algoritmo di apprendimento del perceptron abbia suscitato un tale clamore.

Ma cosa significano questi termini? Cosa sono i "modelli" nei dati? Cosa significa "imparare a riconoscere questi modelli"? incominciamo con l'esaminare questa tabella:

x1	**x2**	**y**
4	2	8
1	2	5
0	5	10
2	1	4

Ciascuna riga della tabella è una terna di valori per le variabili *x1*, *x2* e *y*. In questi dati si nasconde un semplice schema: in ogni riga, il valore di *y* è correlato ai valori corrispondenti di *x1* e *x2*. Vedete se riuscite a individuarlo prima di continuare a leggere.

In questo caso, con carta, matita e un po' di impegno si può capire che *y* è uguale a *x1* più due volte *x2*:

$$y = x1 + 2x2$$

Una piccola precisazione sulla notazione: Faremo a meno del segno di moltiplicazione ("×") tra due variabili o tra una costante e una variabile. Per esempio, scriveremo:

$$2 \times x2 \text{ come } 2x2 \quad \text{e} \quad x1 \times \text{x2 come } x1x2$$

Idealmente, dovremmo scrivere 2*x*2 come $2x_2$ e *x1x2* come x_1x_2, ovvero con i pedici. Ma faremo a meno anche dei pedici, a meno che non sia assolutamente necessario usarli. I puristi storceranno il naso, ma questo metodo ci aiuta a mantenere il testo meno ingombrante e più facile da leggere. Quando incontriamo dei pedici, leggiamo x_i come "x pedice i". Quindi, teniamolo a mente: se incontriamo una lettera, per esempio "*x*", seguita da un numero, per esempio "2", ovvero *x2,* prendiamo l'intero simbolo come un tutt'uno. Se un simbolo (per esempio *x* o *x2*) è preceduto da un numero (per esempio 9) o da un altro simbolo (per esempio, *w1*), allora il numero e il simbolo, o i due simboli, vengono moltiplicati fra loro. Quindi:

$$2x2 = 2 \times x2$$

$$x1x2 = x1 \times x2$$

$$w2x1 = w2 \times x1$$

Tornando alla nostra equazione $y = x1 + 2x2$, possiamo scriverla, più in generale:

$$y = w1x1 + w2x2, \text{ dove } w1 = 1 \text{ e } w2 = 2$$

Per essere chiari, abbiamo trovato solo una delle tante relazioni possibili tra *y, x1* e *x2*. Possono essercene altre. In effetti, per questo esempio, ce ne sono, ma non è il caso di preoccuparsene. Trovare dei modelli non è affatto così semplice come sembrerebbe da questo esempio, ma procediamo con la trattazione.

Abbiamo identificato una "relazione lineare" tra *y,* da un lato, e *x1* e *x2,* dall'altro. "Lineare" significa che *y* dipende solo da *x1* e *x2,* e non da *x1* o *x2* elevati a qualche potenza, o da qualsiasi prodotto di

x1 e *x2*. Inoltre, stiamo usando i termini "equazione" e "relazione" in maniera intercambiabile.

La relazione tra *y, x1* e *x2* è definita dalle costanti *w1* e *w2*. Queste costanti sono chiamate coefficienti, o pesi, dell'equazione lineare che collega *y* con *x1* e *x2*. In questo caso molto semplice, supponendo che esista una relazione lineare, abbiamo ricavato i valori di *w1* e *w2* dopo aver osservato i dati. Ma, spesso, la relazione tra *y* e (*x1, x2, …*) non è così semplice, soprattutto quando si estende a più valori sul lato destro dell'equazione. Per esempio, si consideri:

$$y = w1x1 + w2x2 + w3x3 + \ldots + w9x9$$

oppure, più in generale, per un insieme di *n* pesi e utilizzando una notazione matematica formale:

$$y = w1x1 + w2x2 + w3x3 + \cdots + wnxn = \sum_{i=1}^{n} wixi$$

L'espressione a destra, che utilizza la notazione "sigma", rappresenta la somma di tutti i termini *wixi,* dove *i* assume valori compresi tra 1 e *n*.

Nel caso di 9 coefficienti, sarebbe difficile dedurre i valori *w1, w2, …, w9* solamente osservando visivamente i dati e facendo un po' di aritmetica mentale. È qui che entra in gioco l'*apprendimento*. Se esiste un modo per determinare algoritmicamente i coefficienti, allora l'algoritmo "impara" tali coefficienti. Ma a che scopo farlo?

Una volta appresi i pesi (per esempio *w1* e *w2* nel caso dell'equazione più semplice), dato un valore di *x1* e di *x2*, che non era presente nel nostro set di dati iniziale, possiamo calcolare il valore di *y*. Diciamo che $x1 = 5$ e $x2 = 2$. Inserendo questi valori nell'equazione $y = x1 + 2x2$, si ottiene $y = 9$.

Cosa c'entra tutto questo con la vita reale? Prendiamo un problema molto semplice, pratico e, secondo alcuni, molto noioso. Supponiamo che *x1* rappresenti il numero di camere da letto di una casa, che *x2* rappresenti la metratura totale e che *y* rappresenti il prezzo della casa. Supponiamo che esista una relazione lineare tra (*x1, x2*) e *y*. Allora, apprendendo i pesi dell'equazione lineare dai dati riguardanti le case e i loro prezzi, abbiamo di fatto costruito un modello molto semplice con cui prevedere il prezzo di una casa, dato il numero di camere da letto e la metratura.

L'esempio di cui sopra, che è veramente molto semplice, è un abbozzo di apprendimento automatico. Quello che abbiamo appena

fatto è una forma semplicistica di ciò che viene chiamato "apprendimento supervisionato". Vengono forniti campioni di dati, nei quali è nascosta una correlazione tra una serie di input e una serie di output. Tali dati sono detti annotati, etichettati, o anche "dati di training" (addestramento). Ogni ingresso (*x1,* x2, …, *xn*) ha un'etichetta *y* ad esso associata. Quindi, nella nostra tabella numerica precedente, la coppia di numeri (4, 2) è etichettata con $y = 8$, la coppia (1, 2) con 5 e così via. Abbiamo capito la correlazione. Una volta appresa, possiamo utilizzarla per fare previsioni su nuovi input che non facevano parte dei dati di training.

Inoltre, abbiamo svolto un tipo molto particolare di "problem solving" chiamato "regressione", nel quale, date alcune variabili indipendenti (*x1, x2*), abbiamo costruito un modello (o equazione) per prevedere il valore di una variabile dipendente (*y*).

Esistono molti altri tipi di modelli che avremmo potuto costruire. Ne parleremo a tempo debito.

In questo caso la correlazione, o modello, era così semplice che è stata sufficiente una piccola quantità di dati di addestramento. Ma i moderni metodi di machine learning (ML) richiedono ordini di grandezza molto maggiori e la disponibilità di tali dati è stato uno dei fattori che hanno alimentato la rivoluzione dell'AI.

Gli anatroccoli, invece, si dedicano a una forma ancor più sofisticata di apprendimento. Nessuna anatra genitrice se ne sta seduta a etichettare i dati per i propri anatroccoli, eppure i piccoli imparano. Ma come fanno? Non lo sappiamo, ma capendo come imparano le macchine, forse un giorno potremo comprendere appieno il modo in cui gli anatroccoli e, di fatto, anche gli esseri umani imparano.

Può sembrare inverosimile, ma questo primo passo che abbiamo fatto utilizzando un esempio molto semplice di apprendimento supervisionato ci porta sulla strada verso la creazione di un *sistema di apprendimento supervisionato*. Approfondiremo le moderne "reti neurali profonde", o "deep neural network" (DNN), naturalmente un passo alla volta, con piccoli, anche se talvolta non molto delicati, accenni su vettori, matrici, algebra lineare, probabilità e statistica, calcolo infinitesimale e teoria dell'ottimizzazione. Il perceptron, o percettrone, di Rosenblatt, che abbiamo incontrato brevemente nel prologo, era per l'epoca un esempio sorprendente di uno di questi algoritmi di apprendimento. Poiché era modellato sul modo in cui i neuroscienziati pensavano che funzionassero i neuroni umani, era intriso di misticismo, ma anche della speranza che, un giorno, i perceptron avrebbero davvero fatto fruttare le promesse dell'intelligenza artificiale.

Il primo neurone artificiale

Le radici del perceptron affondano in un articolo del 1943, redatto in seguito a un improbabile incontro tra un neuroscienziato-filosofo e un adolescente senzatetto. Warren McCulloch era un neurofisiologo americano con una formazione filosofica, psicologica e medica. Negli anni Trenta si occupò di neuroanatomia, creando mappe di connettività tra le varie parti del cervello delle scimmie. Nel farlo, si occupò anche della "logica del cervello". A quel punto, il lavoro di matematici e filosofi come Alan Turing, Alfred North Whitehead e Bertrand Russell stava suggerendo l'esistenza di una profonda connessione tra calcolo e logica.

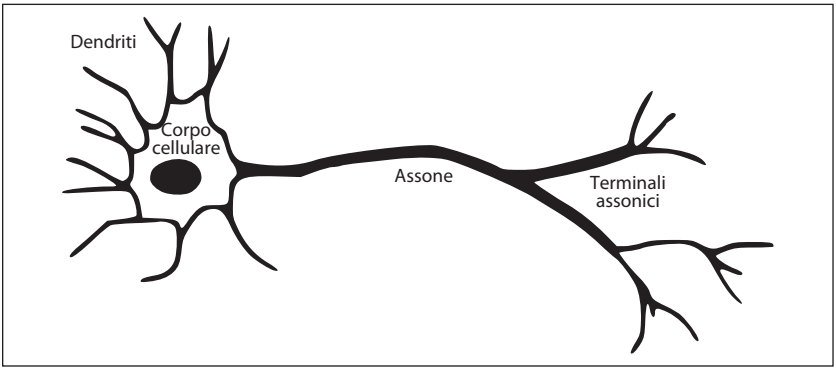
L'affermazione "Se P è vero e Q è vero, allora S è vero" è un esempio di proposizione logica. L'assunzione più diffusa era che tutta la computazione poteva essere ridotta a questo tipo di logica. La domanda che assillava McCulloch era quindi la seguente: se il cervello è un dispositivo computazionale, come molti pensano, come fa a implementare detta logica?

Con questi interrogativi in mente, McCulloch si trasferì nel 1941 dall'Università di Yale all'Università dell'Illinois, dove incontrò un adolescente prodigioso, con tanto talento, di nome Walter Pitts. Il giovane, già un logico affermato e "un protetto dell'eminente logico matematico Rudolf Carnap", frequentava i seminari tenuti dal fisico matematico ucraino Nicolas Rashevsky, a Chicago. Pitts, tuttavia, era un "adolescente confuso, di fatto fuggito da una famiglia che non riusciva ad apprezzare il suo genio". McCulloch e sua moglie, Rook, diedero a Walter una casa. "Ne seguirono interminabili serate seduti intorno al tavolo della cucina dei McCulloch, per cercare di capire come funzionasse il cervello, mentre la figlia dei McCulloch, Taffy, faceva dei piccoli disegni", scrisse l'informatico Michael Arbib. I disegni di Taffy sarebbero in seguito diventate le illustrazioni presenti nell'articolo di McCulloch e Pitts, del 1943: "A Logical Calculus of the Ideas Immanent in Nervous Activity".

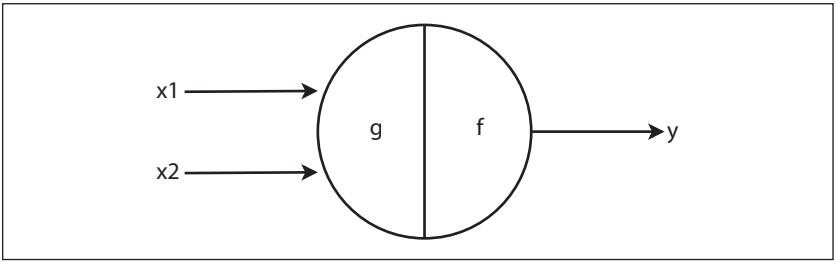
In quel lavoro, McCulloch e Pitts proposero un modello semplificato per i neuroni biologici. Nella figura che segue vediamo una rappresentazione di un neurone generico.

Il corpo cellulare del neurone riceve gli input attraverso le sue ramificazioni, chiamate dendriti. Il corpo cellulare esegue alcuni calcoli su questi input. Poi, in base ai risultati, può inviare un impulso elettrico lungo una proiezione, relativamente lunga, chiamata assone. Il segnale viaggia lungo l'assone e raggiunge le ramificazioni dette terminali assonici, dove viene comunicato ai dendriti dei neuroni vicini.

E così via. I neuroni così interconnessi formano una rete neurale biologica.



McCulloch e Pitts tradussero tutto ciò in un semplice modello computazionale: un neurone artificiale. E mostrarono come, utilizzando uno di questi neuroni artificiali, o neurodi ("neuroni" + "nodi"), si potevano eseguire alcune operazioni logiche booleane di base, come AND, OR, NOT e così via, che sono gli elementi costitutivi del calcolo digitale. Per alcune operazioni booleane, come l'OR esclusivo (XOR) sono necessari più neurodi, ma di questo parleremo più avanti. Quella che segue è l'immagine di un singolo neurodo. Per ora ignoriamo le lettere "g" e "f": ci torneremo fra poco.



In questa semplice versione del modello McCulloch-Pitts (MCP), *x1* e *x2* possono valere 0 o 1. In notazione formale, possiamo dire che:

$$x1, x2 \in \{0, 1\}$$

Questo va letto come "*x1* e *x2* appartengono all'insieme di valori 0 1"; *x1* e *x2* possono assumere solo i valori 0 o 1: nient'altro. L'uscita *y* del neurodo viene calcolata sommando prima gli ingressi e

poi controllando se la somma è maggiore o uguale a una certa soglia, *theta* (θ). In caso affermativo, *y* è uguale a 1; in caso contrario, *y* è uguale a 0.

$$somma = x1 + x2$$

$$Se\ somma \geq \theta: y = 1$$

$$Altrimenti: y = 0$$

Generalizzando al caso di una sequenza arbitraria di ingressi, *x1*, *x2*, *x3*, …, *xn*, possiamo scrivere la formulazione matematica formale del neurodo semplice. Per prima cosa definiamo la funzione *g*(*x*) (si legge "g di x"), che effettua la somma degli ingressi *x1*, *x2*, *x3*, …, *xn*, rappresentati complessivamente dalla "*x*" fra parentesi. Poi definiamo la funzione *f*(*g*(*x*)) (si legge "f di g di x"), che prende la somma ed esegue la soglia per generare l'output, y: è pari a zero se *g*(*x*) è minore di un certo valore θ ed è pari a 1 se *g*(*x*) è maggiore o uguale a θ.

$$g(x) = x1 + x2 + x3 + \cdots + xn = \sum_{i=1}^{n} xi$$

$$f(z) = \begin{cases} 0, & z < \theta \\ 1, & z \geq \theta \end{cases}$$

$$y = f(g(x)) = \begin{cases} 0, & g(x) < \theta \\ 1, & g(x) \geq \theta \end{cases}$$

Con un neurone artificiale come quello descritto, possiamo progettare alcune delle porte logiche booleane di base (AND e OR, per esempio). In un gate logico AND, l'uscita *y* vale 1 se *x1* e *x2* sono entrambi uguali a 1; altrimenti l'uscita vale 0. In questo caso, θ = 2 è sufficiente: l'uscita *y* sarà pari a 1 solo quando *x1* e *x2* sono entrambi 1 (solo così *x1* + *x2* sarà maggiore o uguale a 2). Giocando con il valore di θ possiamo ottenere le altre porte logiche. Per esempio, in una porta OR l'uscita deve essere 1 se *x1* o *x2* sono 1; altrimenti, l'uscita deve essere 0. Quale deve essere il valore di θ?

Il relativamente semplice modello MCP può essere esteso: basta aumentare il numero di ingressi. È possibile imporre che gli ingressi

siano "inibitori", ovvero che *x1* o *x2* siano pari a –1. Se uno degli ingressi del neurodo è inibitorio e si imposta la soglia in modo appropriato, allora il neurodo darà sempre output pari a 0, indipendentemente dal valore di tutti gli altri input. Questo ci permette di costruire delle logiche più complesse. Lo stesso vale per l'interconnessione di più neurodi, in modo che l'uscita di un neurodo funga da ingresso in un altro.

Tutto questo era sorprendente, ma allo stesso tempo limitato. Il neurone McCulloch-Pitts (MCP) è un'unità di calcolo e si possono usare combinazioni diverse per creare qualsiasi tipo di logica booleana. Dato che una computazione digitale è scomponibile in una sequenza di operazioni logiche, è possibile mescolare e abbinare i neuroni MCP per eseguire qualsiasi calcolo. Si trattava di un'affermazione straordinaria, nel 1943. Le radici matematiche dell'articolo di McCulloch e Pitts erano evidenti. L'articolo conteneva solo tre riferimenti: *The Logical Syntax of Language* di Carnap, *Foundations of Theoretical Logic* di David Hilbert e Wilhelm Ackermann e *Principia Mathematica* di Whitehead e Russell. Nessuno di questi aveva a che fare con la biologia. Non c'era alcun dubbio sui risultati rigorosi descritti nell'articolo di McCulloch e Pitts. Eppure, il risultato era semplicemente una macchina in grado di calcolare, non di imparare. In particolare, il valore θ doveva essere definito a mano: il neurone non era in grado di esaminare i dati e di dedurre θ.

Non c'è da stupirsi se il perceptron di Rosenblatt fece così scalpore. Poteva imparare i valori dei suoi pesi partendo dai dati. I pesi codificavano alcune conoscenze, per quanto minime, riguardo agli schemi presenti in tali dati.

Imparare dagli errori

Rosenblatt lasciava spesso i suoi studenti a bocca aperta. George Nagy, che nel 1960 venne alla Cornell University di Ithaca, New York, per conseguire il dottorato con Rosenblatt, ricorda una passeggiata durante la quale i due parlarono della visione stereo. Rosenblatt stupì Nagy per la sua padronanza dell'argomento. "Era difficile non sentirsi un po' ingenui parlando con lui", disse Nagy, ora professore emerito al Rensselaer Polytechnic Institute di Troy, New York; la grande erudizione di Rosenblatt era accentuata dalla sua relativa giovinezza: aveva solo dieci anni più di Nagy.

L'età di Rosenblatt mise quasi in difficoltà i due durante un viaggio. Lui e Nagy dovevano andare da Ithaca a Chicago, per una conferenza. Rosenblatt non aveva ancora scritto la relazione che voleva pre-

sentare, così chiese a Nagy di mettersi al volante, mentre lui avrebbe lavorato. Nagy non aveva mai posseduto un'auto e sapeva a malapena guidare, ma accettò comunque. "Purtroppo, senza accorgermi, rimasi per alcuni chilometri a cavallo di due corsie e così un poliziotto ci fermò", mi raccontò Nagy. Rosenblatt disse al poliziotto che era un professore e che aveva chiesto al suo studente di guidare. Il poliziotto si mise a ridere e disse: "Lei non è un professore, è uno studente". Fortunatamente, Rosenblatt aveva con sé abbastanza documenti da convincere il poliziotto delle sue credenziali e alla fine il tutore della legge li lasciò andare. Rosenblatt guidò per il resto della strada fino a Chicago, dove rimase sveglio tutta la notte per terminare la sua relazione, che presentò il giorno successivo. "Era in grado di fare queste cose", mi disse Nagy.

Quando Nagy arrivò alla Cornell, Rosenblatt aveva già costruito il Perceptron Mark I; abbiamo visto nel prologo che Rosenblatt lo aveva terminato nel 1958, con tanto di articolo sul *New York Times*. Nagy iniziò così a lavorare alla macchina successiva, chiamata Tobermory (che prende il nome dal gatto parlante creato da H. H. Munro, anche detto "Saki"), una rete neurale hardware progettata per il riconoscimento vocale. Nel frattempo, il Perceptron Mark I e le idee di Rosenblatt avevano già ricevuto molta attenzione.

Nell'estate del 1958, l'editore della rivista Research Trends del Cornell Aeronautical Laboratory aveva dedicato un intero numero a Rosenblatt ("a causa dell'insolito significato dell'articolo del Dott. Rosenblatt", scrisse l'editore). L'articolo era intitolato "The Design of an Intelligent Automaton: Introducing the Perceptron – A Machine that Senses, Recognizes, Remembers, and Responds Like the Human Mind". Rosenblatt si sarebbe poi pentito di aver scelto il termine "perceptron" per descrivere il suo lavoro. "È diventato uno dei grandi rimpianti di Rosenblatt, quello di aver usato una parola che suona come una macchina", mi disse Nagy. Con "perceptron", Rosenblatt intendeva in realtà una classe di modelli del sistema nervoso dedicati alla percezione e alla cognizione.

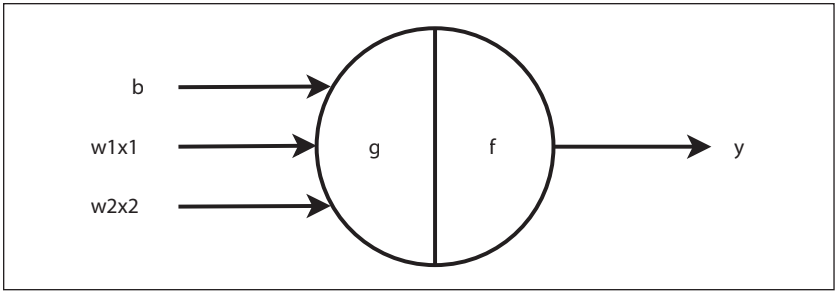
La sua enfasi sul cervello non era casuale. Rosenblatt aveva studiato con James Gibson, uno dei giganti della percezione visiva. Si ispirava anche a McCulloch e Pitts e a Don Ald Hebb, uno psicologo canadese che nel 1949 introdusse un modello su come apprendono i neuroni biologici. Per essere chiari, per "apprendimento" qui intendiamo il riconoscimento di schemi nei dati, non il tipo di apprendimento che di solito associamo alla cognizione umana di alto livello. "Parlava sempre bene di loro", raccontò Nagy.

Mentre McCulloch e Pitts avevano sviluppato dei modelli per i neuroni, le reti di questi neuroni artificiali non erano in grado di ap-

prendere. Nel contesto dei neuroni biologici, Hebb aveva proposto un meccanismo di apprendimento che spesso viene descritto, in qualche misura erroneamente, con la frase: "i neuroni che si accendono insieme si collegano fra loro". Più precisamente, secondo questa interpretazione, il nostro cervello impara perché le connessioni tra i neuroni si rafforzano quando l'uscita di un neurone è costantemente coinvolta nell'accensione di un altro neurone e si indeboliscono quando ciò non avviene. Questo processo è chiamato "apprendimento hebbiano". Fu proprio Rosenblatt a prendere il lavoro di questi pionieri e a sintetizzarlo in una nuova idea: i neuroni artificiali che si riconfigurano durante l'apprendimento, incorporando le informazioni nella forza delle loro connessioni.

Come psicologo, Rosenblatt non aveva accesso alla potenza informatica necessaria per implementare le proprie idee grazie all'elettronica, con hardware e/o software. Così, prese in prestito l'IBM 704 del Cornell Aeronautical Laboratory, un colosso di cinque tonnellate, grande quanto una stanza. La collaborazione si rivelò fruttuosa quando il lavoro di Rosenblatt attirò l'attenzione dei fisici, portando alla pubblicazione di articoli su riviste di psicologia e sull'American Physical Society.

Alla fine Rosenblatt costruì il Perceptron Mark I. Il dispositivo era dotato di una fotocamera che produceva immagini di 20×20 pixel. Il Mark I, quando gli venivano mostrate tali immagini, era in grado di riconoscere le lettere dell'alfabeto. Ma dire che il Mark I "riconosceva" i caratteri è un errore, specificò Nagy. Dopotutto, i sistemi di riconoscimento ottico dei caratteri, che avevano le stesse capacità, erano già disponibili in commercio già dalla metà degli anni '50. "Il punto è che il Mark I *imparava* a riconoscere i caratteri venendo punito con una scarica elettrica quando sbagliava!", raccontava Nagy nei suoi discorsi.



Ma cos'è esattamente un perceptron e come apprende? Nella sua forma più semplice, un perceptron è un neurone McCulloch-Pitts evoluto, dotato di un algoritmo di apprendimento. Di seguito è raf-

figurato un esempio con due ingressi. Si noti che ogni ingresso viene moltiplicato per il suo peso corrispondente. (C'è anche un input aggiuntivo, b, la cui presenza diventerà presto chiara).

Il calcolo effettuato dal perceptron si svolge in questo modo:

$$somma = w1x1 + w2x2 + b$$

$$Se\ somma > 0: y = 1$$

$$Altrimenti: y = -1$$

Più in generale e in notazione matematica:

$$g(x) = w1x1 + w2x2 + \cdots + wnxn + b = \sum_{i=1}^{n} wixi + b$$

$$f(z) = \begin{cases} -1, z \leq 0 \\ 1, z > 0 \end{cases}$$

$$y = f(g(x)) = \begin{cases} -1, g(x) \leq 0 \\ 1, g(x) > 0 \end{cases}$$

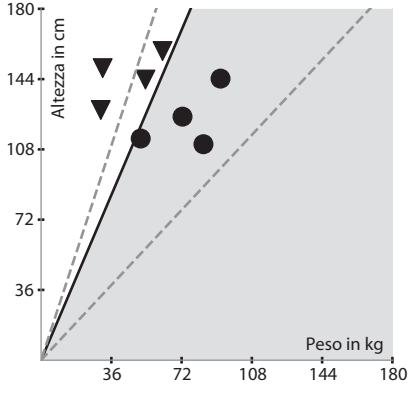
La differenza principale rispetto al modello MCP descritto in precedenza è che gli ingressi del perceptron non devono essere necessariamente binari (0 o 1), ma possono assumere qualsiasi valore. Inoltre, questi input vengono moltiplicati per i loro pesi corrispondenti, per cui ora abbiamo una somma ponderata. A ciò si aggiunge un ulteriore termine *b,* il bias. L'uscita, *y,* è –1 o +1 (invece di 0 o 1, come nel caso del neurone MCP). A differenza del neurone MCP, il perceptron può apprendere il valore corretto dei pesi e del bias per la risoluzione di un qualche problema.

Per capire come funziona, si consideri un perceptron che cerca di classificare una persona come obesa, $y = +1$, o come non obesa, $y = -1$. Gli input sono il peso corporeo della persona, *x1,* e l'altezza, *x2*. Supponiamo che il set di dati contenga un centinaio di voci, ognuna delle quali comprende il peso corporeo di un individuo, la sua altezza e un'etichetta che dice se un medico ritiene che sia obeso, secondo le linee guida stabilite dal National Heart, Lung, and Blood Institute. Il

compito di un perceptron è quello di apprendere i valori di *w1* e *w2* e il valore del termine di bias *b,* in modo da classificare correttamente ogni persona dell'insieme di dati come "obesa" o "non obesa". Nota: stiamo analizzando il peso corporeo e l'altezza di una persona, e allo stesso tempo parliamo dei pesi del perceptron (*w1* e *w2*); teniamo a mente questi due diversi significati della parola "peso" durante la lettura. Una volta che il perceptron ha appreso i valori corretti di *w1* e *w2* e del bias, è pronto a fare previsioni. Dato il peso corporeo e l'altezza di una persona *non presente* nel set di dati originale (quindi non si tratta di consultare una semplice tabella di voci), il perceptron è in grado di classificare la persona come obesa o non obesa. Naturalmente, alla base di questo modello ci sono alcune ipotesi, in gran parte legate alle distribuzioni di probabilità, di cui parleremo nei capitoli successivi. Ma il perceptron fa un'assunzione di base: presuppone che esista una divisione netta e chiara tra le categorie di persone classificate come obese e quelle classificate come non obese.

Nel contesto di questo semplice esempio, se si dovessero tracciare i pesi e le altezze corporee delle persone su un grafico xy, con i pesi sull'asse delle ascisse e le altezze sull'asse delle ordinate, in modo che ogni persona sia un punto sul grafico, l'ipotesi di "netta separazione" è equivalente ad affermare che esiste una linea retta che divide i punti che rappresentano gli obesi dai punti che rappresentano i non obesi. In tal caso, si dice che l'insieme dei dati è "linearmente separabile".

Ecco un grafico di ciò che accade quando il perceptron apprende. Partiamo da due serie di punti di dati, una rappresentata con dei cerchi ($y = +1$, obesi) e l'altra con dei triangoli ($y = -1$, non obesi).



Ogni punto è caratterizzato da una coppia di valori (*x1, x2*), dove *x1* è il peso corporeo della persona in chilogrammi, tracciato lungo

l'asse delle ascisse, e *x2* è l'altezza in centimetri, tracciata lungo l'asse delle ordinate.

Il perceptron inizia con i suoi pesi, *w1* e *w2*, e il suoi bias inizializzati a zero. I pesi e il bias rappresentano una linea sul piano xy. Il perceptron cerca poi di trovare una linea di separazione, definita da un certo insieme di valori per i pesi e il bias, che permetta di classificare i punti. All'inizio classifica alcuni punti in modo corretto e altri in modo errato. Due dei tentativi errati sono rappresentati con delle linee grigie tratteggiate. In questo caso, si può notare che, in un tentativo, tutti i punti si trovano su un lato della linea tratteggiata, quindi i triangoli vengono classificati correttamente, ma i cerchi no; e in un altro tentativo i cerchi sono corretti ma alcuni triangoli sono sbagliati. Il perceptron impara dai propri errori e regola i pesi e il bias. Dopo numerosi passaggi attraverso i dati, il perceptron scopre almeno una serie di valori corretti dei suoi pesi e del suo bias. Trova così una linea che delimita i due diversi cluster: i cerchi e i triangoli si trovano su due lati opposti. Tale linea è rappresentata come continua, non tratteggiata, e separa lo spazio delle coordinate in due regioni, una delle quali ombreggiata in grigio. I pesi appresi dal perceptron determinano la pendenza della linea; il bias determina la distanza, o l'offset, della linea dall'origine.

Una volta che il perceptron ha appreso la correlazione tra le caratteristiche fisiche di una persona (peso corporeo e altezza) e il fatto che quella persona sia obesa o no ($y = +1$ o -1), è possibile fargli elaborare il peso corporeo e l'altezza di un individuo i cui dati non sono stati utilizzati durante l'addestramento e il perceptron ci dirà se debba essere classificato come obeso. Naturalmente, a quel punto il perceptron farà la sua migliore previsione, avendo imparato i suoi pesi e il suo bias, ma la previsione potrà essere sbagliata. Riusciamo a capire perché? Proviamo a individuare il problema semplicemente osservando il grafico. Suggerimento: quante linee diverse riusciamo a tracciare, tali che separino i cerchi dai triangoli? Come vedremo, gran parte dell'apprendimento automatico si riduce alla minimizzazione dell'errore di predizione.

Quella descritta sopra è una singola unità di perceptron, ovvero un neurone artificiale. Sembra molto semplice e ci si può chiedere il motivo di tanto clamore. Ebbene, immaginate se il numero di ingressi nel perceptron fosse superiore a due: (*x1*, *x2*, *x3*, *x4*, e così via), e ciascun ingresso *xi* avesse il proprio asse di riferimento. Non sarebbe possibile effettuare una semplice procedura aritmetica mentale e risolvere il problema. Infatti, non è più sufficiente una linea per separare i due cluster, che esisterebbero in dimensioni molto superiori alle semplici due. Per esempio, quando si hanno tre punti (*x1*, *x2*, *x3*), i dati sono tridimensionali: è necessario un piano 2D per separare i

punti dei dati. In dimensioni pari o superiori a quattro, è necessario un iperpiano (che non possiamo visualizzare con la nostra mente 3D). In generale, l'equivalente dimensionale di una retta 1D o di un piano 2D è chiamato, appunto, iperpiano.

Ripensiamo al 1958. Rosenblatt costruì il suo Perceptron Mark I con numerose unità del tipo sopra esposto. Poteva elaborare un'immagine di 20×20 pixel, per un totale di 400 pixel, e ogni pixel corrispondeva a un valore x di input. Quindi, il Mark I accettava in ingresso una lunga serie di valori: $x1$, $x2$, $x3$, …, $x400$. Una complessa disposizione di neuroni artificiali, sia con pesi fissi e casuali, sia con pesi che potevano essere appresi, trasformava questo vettore di 400 valori in un segnale di uscita che poteva essere usato per discernere il modello nell'immagine. Questa che ho scritto è una descrizione eccessivamente semplificata: alcune operazioni di calcolo erano abbastanza complesse da rendere necessario l'utilizzo di un IBM 704 (nel Capitolo 10 illustreremo alcuni dettagli architetturali di tale computer). Il Mark I poteva imparare a classificare le lettere dell'alfabeto codificate nei valori dei pixel. Tutta la logica appena descritta, scalata per gestire ben 400 ingressi, era stata sviluppata come puro hardware. La macchina, una volta terminato l'apprendimento, immagazzinava la conoscenza nella forza (pesi) delle sue connessioni. Non c'è da stupirsi se molti diedero sfogo alla loro immaginazione.

Ma se si esamina da vicino ciò che il perceptron è in grado di imparare, i suoi limiti, ovviamente con il senno di poi, diventano evidenti. L'algoritmo permette al perceptron di apprendere le correlazioni tra i valori di ($x1$, $x2$, …, $x400$) e il corrispondente valore di y, se tali correlazioni esistono nei dati. Certo, le impara senza che gli venga detto esplicitamente cosa sono, ma si tratta comunque solo di correlazioni. Identificarne una è equivalente a pensare e ragionare? Sicuramente, se il Mark I riesce a distinguere la lettera "B" dalla lettera "G", si basa semplicemente su modelli e non attribuisce a quelle lettere un significato tale da indurre un ulteriore ragionamento. Queste domande sono al centro del moderno dibattito sui limiti delle "reti neurali profonde", o "deep neural networks", i sorprendenti discendenti dei percettori. Esiste un percorso ben preciso che collega quei primi percettori alla tecnologia dei "modelli linguistici di grandi dimensioni", o "large language models" (LLM), e all'intelligenza artificiale utilizzata, per esempio, sulle auto a guida autonoma. Non è un percorso rettilineo: è piuttosto lungo e tortuoso, con svolte improvvise e vicoli ciechi. Ma è comunque una via affascinante e stimolante, che stiamo per percorrere.

La costruzione del dispositivo perceptron fu un risultato molto importante. Ma ancora più importante fu la dimostrazione matema-

tica che un singolo strato di perceptron troverà sempre un iperpiano di separazione lineare, se i dati sono linearmente separabili. Per poter comprendere la dimostrazione si rende necessario un piccolo approfondimento sui vettori e su come essi costituiscano l'ossatura dei metodi utilizzati per rappresentare i dati nell'ambito dell'apprendimento automatico, o "machine learning".

È il nostro primo pit-stop matematico.