Marco Maggesi

Una introduzione visiva al Machine Learning e al Deep Learning

Laboratorio dell'ISPF, XIX, 2022

[3]
DOI: 10.12862/Lab22MGM

Introduzione

Le nuove tecniche dell'Intelligenza Artificiale (IA) stanno diventando pervasive nella nostra esperienza quotidiana. Così come ogni automobilista deve conoscere qualche rudimento sulla meccanica di un autoveicolo (anche colui che non aprirà mai il cofano della sua auto), sarebbe auspicabile che chiunque usi – talvolta inconsapevolmente – un servizio basato sull'IA sia fornito di qualche strumento concettuale per capirne il funzionamento.

Com'è noto, una spiegazione approfondita delle nuove tecnologie di Apprendimento Automatico non è possibile senza alcune nozioni avanzate di Matematica. Nonostante ciò, certe idee fondamentali possono essere rese accessibili in modo abbastanza semplice. Questo, in ogni caso, è il nostro auspicio e l'obiettivo di questo contributo.

Tenteremo di spiegare in che cosa consista l'"apprendimento" a cui si fa riferimento nell'espressione *Machine Learning* (ML) e quale sia la differenza – o meglio la specificità – del *Deep Learning* (DL), spesso menzionato in questo contesto. Gli strumenti dei quali ci avvarremo in questa breve panoramica saranno principalmente dei simulatori di reti neurali artificiali, che permetteranno, in modo visivo e interattivo, di esplorare, mediante degli esperimenti guidati, alcuni meccanismi fondamentali che contribuiscono a creare l'"intelligenza" di cui si parla.

Reti neurali artificiali

In questa breve nota, ci concentreremo sul Machine Learning basato sulle reti neurali artificiali (ANN – Artificial Neural Networks), una delle tecniche attualmente più adottate. Vale la pena di sottolineare che, seppure il concetto di ANN sia fortemente ispirato dal cervello biologico, le differenze sono tante e profonde, quindi ogni analogia può essere intellettualmente utile e stimolante, ma anche potenzialmente fuorviante.

Da una prospettiva storica, potrebbe sorprendere che questo argomento appaia oggi così innovativo. Infatti, la nozione di neurone artificiale e ANN risale ai primi anni '40 dello scorso secolo[1] e l'idea è stata esaminata a più riprese nei decenni successivi, ottenendo risultati sia teorici che pratici. La svolta recente è dovuta ad alcune novità, sia tecnologiche che matematiche, le quali hanno drasticamente migliorato la capacità di "creare" delle ANN per dei compiti prefissati. Questo procedimento va sotto il nome di "addestramento", perché mira a modificare una data rete neurale affinché svolga sempre meglio un dato compito. Per capire meglio questi concetti, è necessario avere un'idea più precisa di che cosa sia una rete neurale artificiale.

Un neurone artificiale altro non è che una funzione numerica (schematicamente rappresentata in Figura 1), ossia una regola che a certi valori (numeri reali) $x_1, \ldots, x_n$ di ingresso associa un valore y in uscita. Il calcolo svolto è elementare, coinvolgendo essenzialmente solo somme e moltiplicazioni.

[1] W. McCulloch - Walter Pitts, *A Logical Calculus of Ideas Immanent in Nervous Activity*, «Bulletin of Mathematical Biophysics», 5 (4) 1943, pp. 115–133, doi:10.1007/BF02478259.

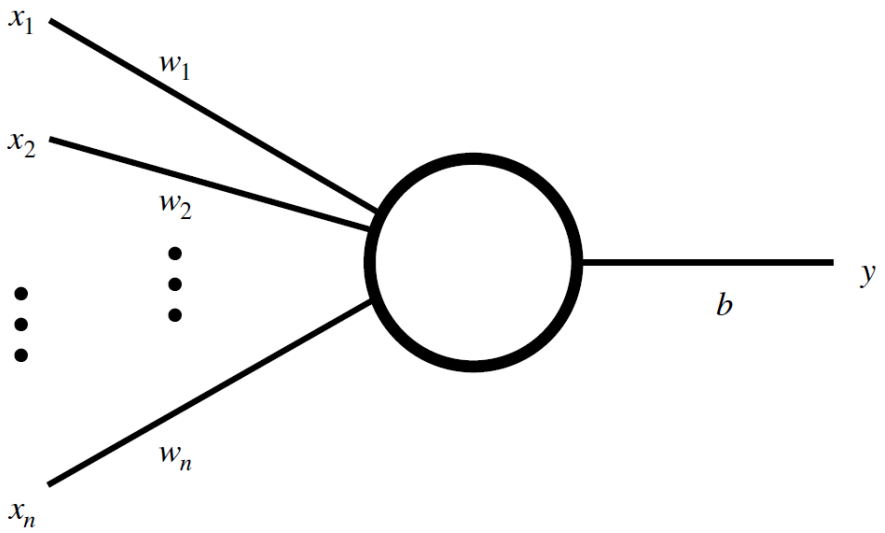

Figura 1. Schema di un neurone artificiale

Quello che conta sapere, per quello che segue, è che il comportamento di un neurone è determinato da certi "parametri". Modificare il neurone significa semplicemente cambiare (aggiustare) tali parametri: l'addestramento consiste essenzialmente nel cercare, attraverso un processo iterativo di approssimazione, valori opportuni per questi parametri per tutti i neuroni di una rete.

Per fare un esempio, qui descriveremo esplicitamente un tipo di neurone molto utilizzato detto ReLU (*Rectified Linear Unit*). Nella figura sono indicati i valori di ingresso e uscita e il cerchio simbolizza l'unità che svolge il calcolo. I parametri sono i valori w_1, …, w_n, detti pesi (*weights*), e b, detta preferenza (*bias*), che determinano il comportamento del neurone. Il calcolo svolto dal "neurone" (ossia dalla funzione matematica) può essere schematizzato in due passi: nel primo passo si calcola il valore intermedio y_0 dato dalla *somma pesata* dei valori di ingresso e b

$$y_0 = w_1x_1 + w_2 x_2 + \cdots + w_nx_n + b,$$

successivamente, nel secondo passo, si prende il massimo y come il massimo tra e zero

$$y = \max(0,y_0).$$

Quando quest'ultimo passaggio non viene effettuato, e si prende semplicemente $y = y_0$, il neurone si dice lineare (LU).

Una rete neurale è formata da vari neuroni connessi tra loro, tipicamente organizzati in strati. In Figura 2 è schematizzato un semplice esempio di rete

con uno strato con tre valori di ingresso x_1, x_2, x_3 che vengono elaborati dai neuroni degli strati interni, per arrivare a fornire due stati di uscita y_1, y_2.

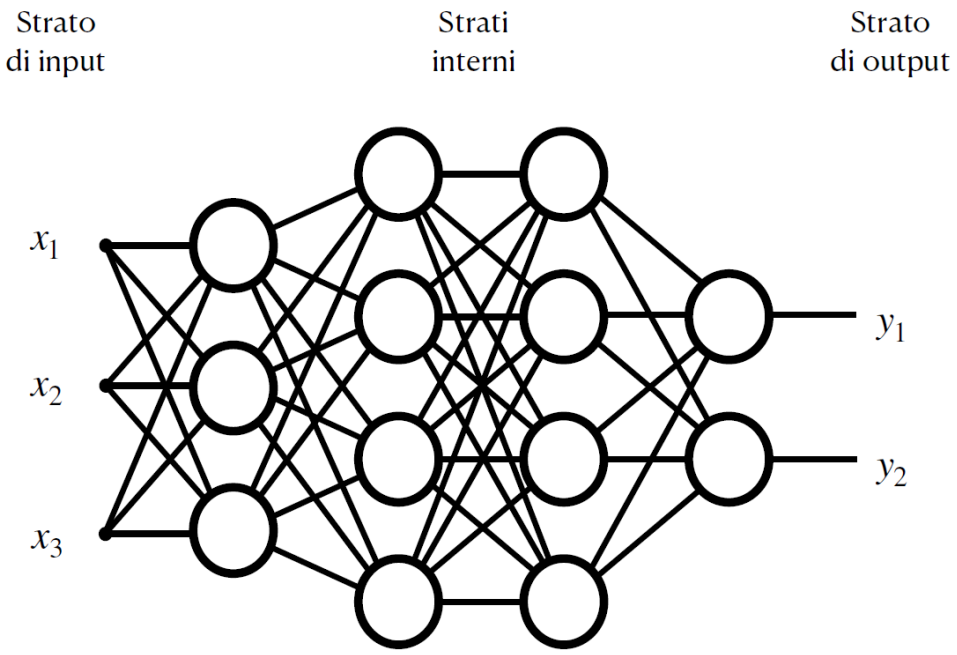

Figura 2. Schema di una semplice rete neurale artificiale

Da un punto di vista puramente formale, questa "rete" non è altro che una descrizione suggestiva di una funzione matematica

$$Y = F_W(X)$$

dove $X = (x_1, x_2, x_3)$ è l'input, $Y = (y_1, y_2)$ è l'output e W è la collezione dei parametri di tutti i neuroni della rete, ossia tutti i valori w_i e b per ogni neurone della rete. Con un po' di pazienza, si possono contare i parametri della rete della figura (il numero di parametri di ogni singolo neurone è uno in più del suo numero di ingressi) e si troverà che si hanno in tutto 58 parametri per questo particolare esempio (7 neuroni con 3 ingressi ciascuno nei due strati di sinistra e 6 neuroni con 4 ciascuno nei due strati di destra), per cui W sarà una lista di 58 valori numerici i quali stabiliscono il comportamento della rete. Più avanti considereremo reti con molti più neuroni e connessioni e, quindi, con molti più parametri. Ad oggi sono state costruite reti aventi fino a centinaia di miliardi di parametri.

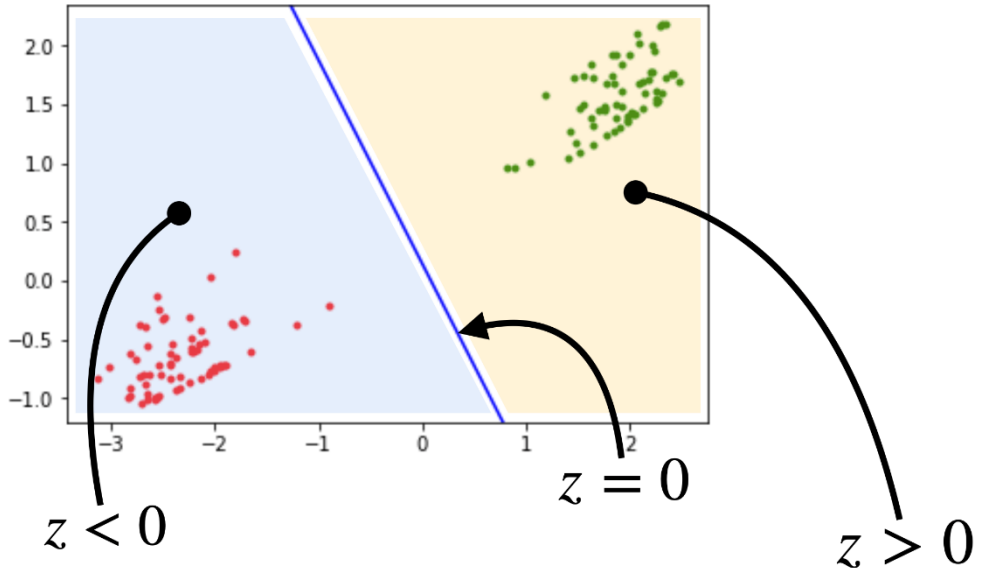

Figura 3. Problema di classificazione

Un semplice problema di classificazione

Fin qui abbiamo descritto sommariamente che cosa sia una rete neurale artificiale, ma non abbiamo ancora spiegato in che modo questa possa essere "addestrata" a svolgere qualche compito. Cominceremo con un esempio elementare. Consideriamo un certo numero di punti sul piano cartesiano. Possiamo immaginare che questi punti indichino dei luoghi di una carta geografica. Avremo due tipi di punti. Nell'esempio in Figura 3 abbiamo un'area con punti rossi in basso a sinistra e una con punti verdi in alto a destra. Guardando la scala graduata a bordo figura, un punto del primo tipo potrebbe avere coordinate ad esempio (−2.3, − 0,7), mentre un punto del secondo tipo potrebbe avere coordinate (1.9,1.6). Date le coordinate (x, y) di un punto vogliamo stabilire se questo sia un punto di tipo rosso o di tipo verde.

Configureremo una rete costituita da un solo neurone per svolgere questo compito, rappresentato in Figura 4. Per semplicità utilizzeremo un neurone lineare (LU) invece che ReLU. Il neurone avrà due ingressi e *y* dati dalle coordinate del punto da esaminare e la sua uscita sarà un numero che indicherà se il punto è nell'area dei punti rossi o in quella dei punti verdi. Ad esempio, possiamo convenire che il neurone produrrà negativo per punti rossi e positivo per quelli verdi. In figura, il luogo dei punti con nullo è indicato dalla retta di separazione blu, e le aree con z positivo e negativo sono rispettivamente quella a sinistra in basso della retta (in azzurro) e quella in alto a destra in giallo. I parametri saranno in tutto tre: due pesi e *b* relativi ciascuno ai due ingressi e *y* e un parametro per la preferenza.

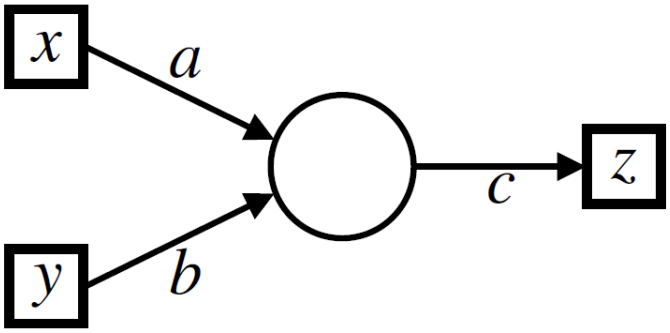

Figura 4. Neurone per il problema di classificazione

La funzione matematica che questo neurone rappresenta è

$$z = F_{a,b,c}(x, y) = a\,x + b\,y + c$$

Perché questo neurone (questa funzione) svolga correttamente i l compito prefissato, dobbiamo cercare dei valori adatti dei parametri a, b, c.

È disponibile online una applicazione interattiva all'indirizzo web <https://www.geogebra.org/classic/bhgpgwdd>[2] che permette di trovare sperimentalmente i valori dei parametri.

L'interfaccia è mostrata in Figura 5. Si possono vedere i due insiemi di punti e in alto a sinistra i valori dei parametri che possono essere variati agendo sui cursori.

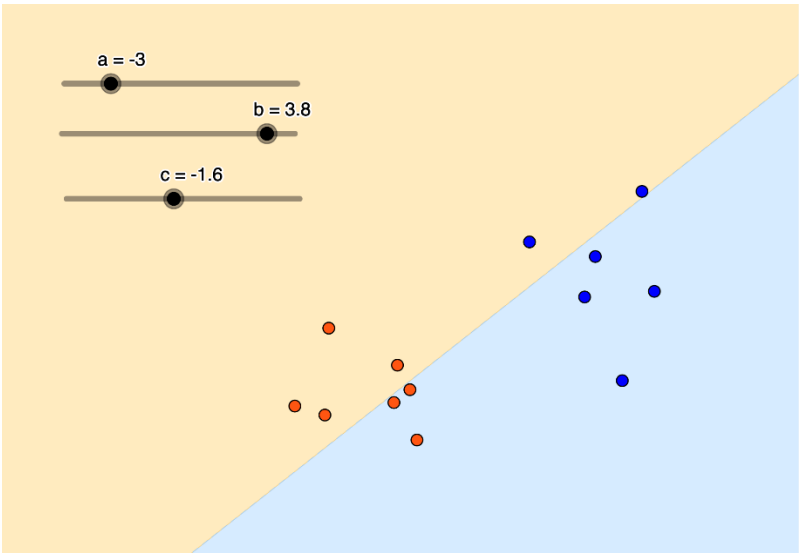

Figura 5. Interfaccia interattiva per la configurazione del neurone

Come per la figura precedente, le due aree gialla e blu indicano dove, con la parametrizzazione data, il neurone individua i due gruppi di punti.

[2] Tutti i collegamenti web indicati in questo articolo sono stati verificati il 6.12.2022

Agendo sui cursori si può configurare il neurone perché le due aree vengano fatte corrispondere ai punti. Fatto questo, abbiamo un neurone che è in grado di svolgere correttamente il suo compito di classificazione.

Il procedimento di addestramento della rete a cui abbiamo già accennato altro non è che rendere automatica (grazie ad alcuni strumenti matematici) questa procedura.

Addestramento automatico di un neurone per il problema di classificazione

Continuiamo la nostra esplorazione usando un altro strumento interattivo, Tensorflow Natural Network Playground, che ci permette di vedere all'opera il processo di apprendimento automatico. La pagina web si trova all'indirizzo <https://playground.tensorflow.org/>. Nel corso della discussione suggeriremo degli indirizzi specifici per accedere a determinate configurazioni.

Cominciamo configurando la rete con un solo neurone e due ingressi, riportato in Figura 6, come già detto nella sezione precedente[3].

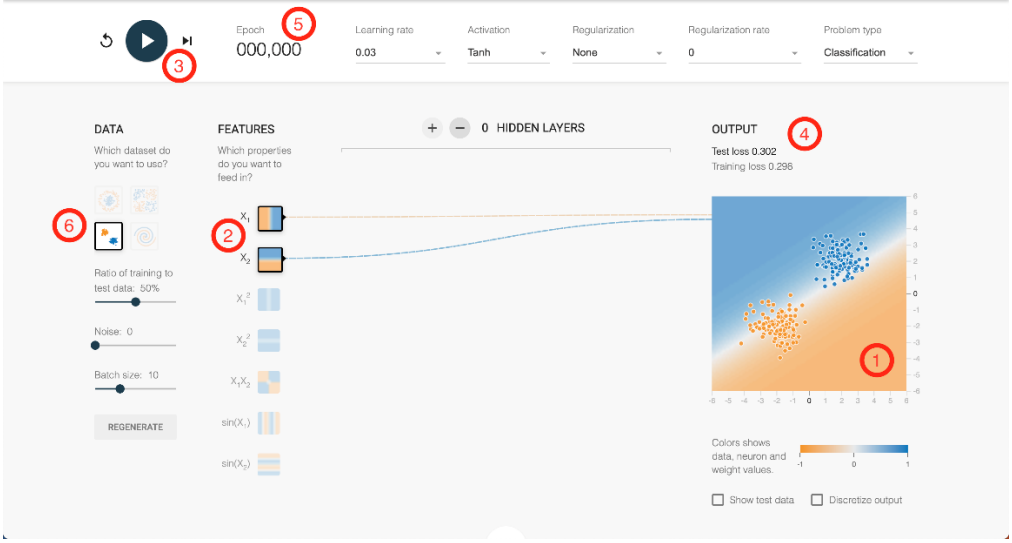

Figura 6. Simulazione dell'addestramento di un neurone con il *Tensorflow Natural Network Playground*

A destra della figura è mostrato l'output del neurone (1) prima dell'inizio della simulazione e, quindi, quando ancora non è configurato correttamente (infatti le aree blu e gialle non sono disposte in modo da accogliere i punti blu e

[3] Questa configurazione è accessibile andando all'indirizzo <https://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=10&dataset=gauss&regDataset=reg-plane&learningRate=0.03&regularizationRate=0.003&noise=0&networkShape=&seed=0.19976&showTestData=false&discretize=false&percTrainData=80&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false>.

gialli rispettivamente). I valori di ingresso del neurone sono rappresentati dai quadrati annotati con x_1 e x_2 in (2).

Cliccando sul pulsante in alto a sinistra (3) si dà inizio all'allenamento della rete. L'animazione mostrerà la rete (il neurone) che "impara" aggiustando i parametri fino a quando le aree blu e gialle non sono collocate in modo ottimale. In alto a destra è riportato il valore del *Training loss* (4), che dà una misura della qualità della classificazione data dalla rete. Più vicino a zero è questo valore e maggiore è la precisione della classificazione fornita dalla rete. Durante l'apprendimento il valore varia e la sua evoluzione è riportata in un grafico, accanto al valore numerico. L'apprendimento è un procedimento iterativo. Le iterazioni di questo processo si chiamano "epoche"; il numero di epoche è riportato in alto a sinistra (5). Il cruscotto presenta vari altri parametri e opzioni che per il momento possono essere ignorati.

L'addestramento ha successo ed è molto rapido in questo caso semplice (Figura 7). Come abbiamo detto, le impostazioni iniziali dei parametri sono casuali, quindi, in linea di principio, si potrebbe ripetere l'esperimento dell'addestramento e ottenere esiti di volta in volta molto diversi. Questo infatti può succedere quando si considerano esempi più complessi, ma non in questo caso. Nelle applicazioni pratiche l'apprendimento può richiedere molti giorni di calcolo e può anche fallire (non produrre una configurazione della rete in grado di svolgere il suo compito). Ad esempio, possiamo cambiare i dati del problema selezionando una delle quattro immagini sotto la scritta "DATA" a sinistra (6). Si osserverà, anche provando più volte, che con le altre tre configurazioni dei dati disponibili il processo non riesce a produrre una configurazione dei parametri (essendo la rete troppo semplice, perché costituita da un solo neurone) che fornisca una buona classificazione.

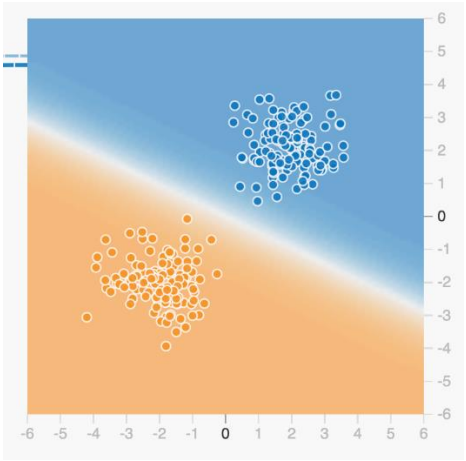

Figura 7. Il neurone alla fine della fase di addestramento

Reti neurali multistrato e Deep Learning

Consideriamo, ad esempio, la configurazione con la corona circolare. Si riesce a capire intuitivamente che, per ottenere una rete classificatrice, non basterà un solo neurone. Proviamo quindi ad aggiungere uno strato interno con due neuroni, come in Figura 8.

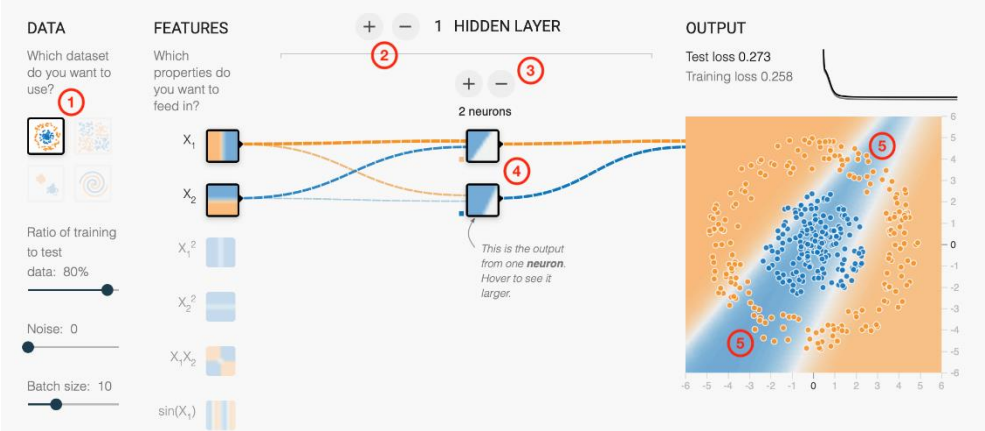

Figura 8. Classificazione con una rete formata da tre neuroni, due interni e uno esterno

Per ottenere questa configurazione[4] si deve selezionare il dataset a sinistra (1) e il numero di strati interni (2) e il numero di neuroni nello strato (3). Il simulatore mostra lo stato del neurone in un quadrato a destra (4). Sperimentalmente si vede bene che anche una rete con tre neuroni (due nello strato interno e uno nello strato di output) non riesce a fornire una buona classificazione. Nella figura si vedono infatti delle regioni in cui i punti non sono classificati correttamente (5).

Facendo qualche prova, si osserva che, in molti casi, può bastare anche aggiungere un solo neurone nello strato intermedio per ottenere una rete che può essere addestrata a risolvere efficacemente questo problema di classificazione (Figura 9).

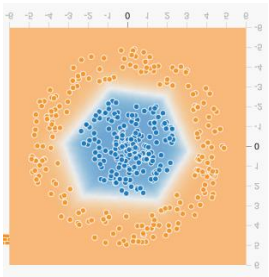

Figura 9. Risultato della classificazione con una rete con 3 neuroni interni

[4] Si può aprire il simulatore questa configurazione direttamente da questo url: <https://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0.003&noise=0&networkShape=2&seed=0.90981&showTestData=false&discretize=false&percTrainData=80&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false>.

Se il dataset è geometricamente più complesso, diventa più difficile ottenere una rete con un solo strato interno capace di fornire una classificazione corretta. Consideriamo ad esempio il dataset a spirale con la configurazione mostrata in Figura 10 nel quale si hanno tre strati interni di 6 neuroni ciascuno[5]. Maggiore è il numero di strati e di neuroni e maggiori sono le potenzialità della rete di poter essere addestrata al problema di classificazione. Allo stesso tempo però, il processo di addestramento è più difficile e computazionalmente costoso.

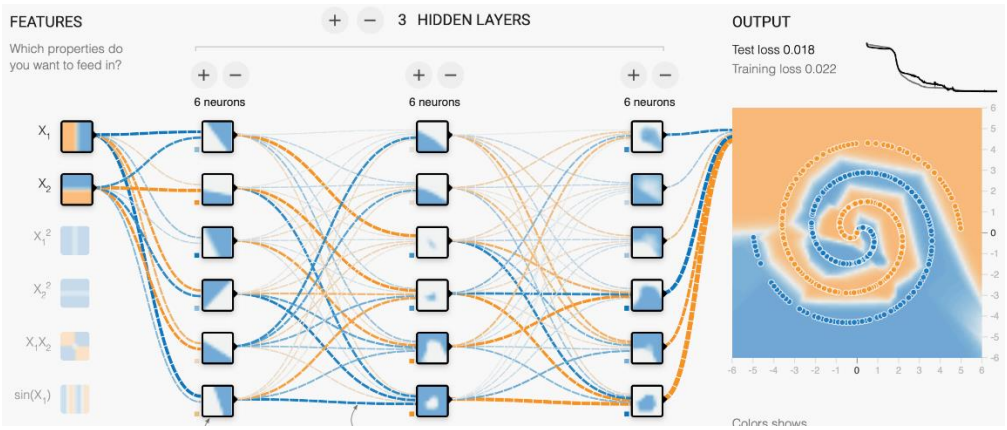

Figura 10. Una rete multistrato per il problema di classificazione del dataset a spirale

Questo mostra intuitivamente che utilizzando reti multistrato è possibile, in linea di principio, costruire dispositivi in grado di risolvere problemi progressivamente più sofisticati. Perseguendo questa tendenza si è condotti a considerare reti "profonde" (cioè con molti strati) ed è questo che si intende per *Deep Learning*. Solo in tempi piuttosto recenti sono stati concepiti metodi abbastanza efficaci per addestrare reti profonde. Ad esempio, è solo nel 2019 che Yoshua Bengio, Geoffrey Hinton e Yann LeCun, ritenuti tre dei più importanti padri fondatori della disciplina, sono stati insigniti del premio Turing "For conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing"[6].

[5] Si può aprire il simulatore questa configurazione direttamente da questo url: <https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=spiral&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=6,6,6&seed=0.18084&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false>.

[6] "Fathers of the Deep Learning Revolution Receive ACM A.M. Turing Award" dal sito web della Association for Computing Machinery — ACM <https://www.acm.org/binaries/content/assets/press-releases/2019/march/turing-award-2018.pdf>.

La difficoltà dell'addestramento

Come anticipato, in linea di principio una rete più complessa (cioè avente un maggior numero di strati e di neuroni) è potenzialmente adatta a compiti più difficili, ma, allo stesso tempo, il suo addestramento è più difficoltoso. Infatti, come già detto, tale processo consiste nel trovare i valori ottimali di certi parametri e maggiore è il numero di tali parametri e più difficile diventa modificarli per ottenere l'effetto voluto. Per fare un esempio con il simulatore, cambiamo l'esperimento precedente aggiungendo ulteriori neuroni e strati intermedi fino al massimo che il simulatore permette (6 strati intermedi di 8 neuroni ciascuno), come visualizzato in Figura 11[7].

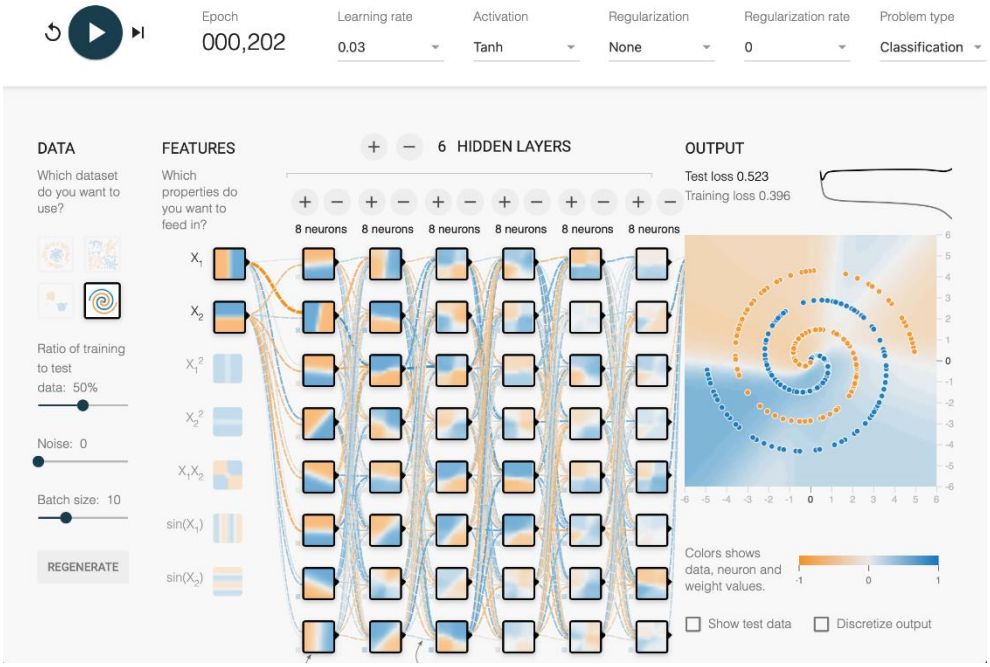

Figura 11. All'aumentare del numero di neuroni e strati intermedi l'addestramento diventa più difficoltoso.

Rispetto alla rete della Figura 10, il numero di neuroni è più che raddoppiato e il numero dei parametri è quasi quattro volte maggiore. Ci si potrebbe quindi aspettare che questa rete possa evolvere per diventare un classificatore molto più preciso dell'esempio precedente. Tuttavia, avviando il simulatore, si osserverà che talvolta, al passare delle epoche, la capacità della rete di risolvere il problema di classificazione non migliora in modo sensibile, anzi, talvolta

[7] Si può aprire il simulatore questa configurazione direttamente da questo url: <https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=spiral&regDataset=regplane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=8,8,8,8,8,8&seed=0.18084&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false>.

peggiora. Questo è reso graficamente evidente dal fatto che le aree blu e arancio restano pallide e diffuse, invece di diventare ben demarcate e seguire l'andamento dei punti del dataset, come mostrato in figura. Quando si considerano reti con migliaia di parametri (o centinaia di milioni), il problema del loro addestramento può essere una vera sfida tecnologica. In questi casi è necessario escogitare, con l'aiuto di strumenti matematici e informatici, strategie specifiche per rendere efficace il processo d'addestramento.

Conclusioni

L'IA porta con sé riflessioni profonde su temi ampi, che vanno dall'impatto socioeconomico alle implicazioni etiche, fino ad interrogarsi sulla natura stessa del pensiero e della coscienza. Questa nota è stata scritta con la convinzione che l'idea delle "macchine che pensano" sia troppo generica e approssimativa – e in una certa misura mistificatrice – per fornire una base a questo tipo di indagini.

Se chiamiamo *smart* i nostri telefoni è perché vorremmo che fossero *intelligenti* (o addirittura *astuti*?), non perché lo siano davvero. In un articolo del 1976 intitolato *Artificial intelligence meets natural stupidity*[8], l'autore Drew McDermott introduce un nome perfetto per questa tendenza a confondere i risultati sperati da quelli effettivamente raggiunti: *wishful mnemonics*. L'Intelligenza Artificiale è pervasa dalla *wishful mnemonics*, a partire dal nome stesso della disciplina. A quasi cinquant'anni di distanza, le considerazioni di McDermott sono ancora attuali.

In questo contesto, la prospettiva di un matematico può offrire un appiglio per raggiungere una concettualizzazione alternativa, forse più equilibrata, dell'argomento. Nel caso specifico del *Machine Learning*, questo ci porta a riformulare le idee fondamentali come segue: le reti neurali (artificiali) sono semplicemente delle *funzioni (*matematiche) e il tipo di "intelligenza" (artificiale) che realizzano non è altro che un *meccanismo di approssimazione*; l'"addestramento", quindi, è banalmente un processo di *ottimizzazione*. Gli aspetti tecnici possono essere talvolta difficilmente accessibili ai non specialisti, ma questo punto di vista può essere utile a tutti per avere una prima intuizione sul funzionamento – e quindi sulle effettive potenzialità e i pericoli intriseci – di questa tecnologia.

[8] D. McDermott, *Artificial intelligence meets natural stupidity*, «SIGART Bull.», 57 (April 1976), pp. 4-9, doi: https://doi.org/10.1145/1045339.1045340

Marco Maggesi
Università degli Studi di Firenze
marco.maggesi@unifi.it

– Una introduzione visiva al Machine Learning e al Deep Learning

Citation standard:
MAGGESI, Marco. Una introduzione visiva al Machine Learning e al Deep Learning. Laboratorio dell'ISPF. 2022, vol. XIX [3]. DOI: 10.12862/Lab22MGM.

Online: 31.12.2022

ABSTRACT
A visual introduction to Machine Learning and Deep Learning. We propose a brief introduction to some basic concepts of Machine Learning and Deep Learning with the help of some simulators available on the Web. The resulting images and animations give a visual representation of the operation and training process of an artificial neural network.

KEYWORDS
Artificial Intelligence; Machine Learning; Deep Learning; Neural networks

SOMMARIO
Proponiamo una breve introduzione ad alcuni concetti basilari del Machine Learning e del Deep Learning con l'aiuto di alcuni simulatori disponibili sul web. Le immagini e le animazioni ottenute offrono una rappresentazione visiva del funzionamento e del processo di addestramento di una rete neurale artificiale.

PAROLE CHIAVE
Intelligenza Artificiale; Machine Learning; Deep Learning; Reti neurali