Atti

DELLA ACCADEMIA ROVERETANA DEGLI AGIATI

CCLXXIII ANNO ACCADEMICO
2023 ser. X, vol. V, B

Classe di Scienze matematiche, fisiche e naturali

SCRIPTA EDIZIONI

Diego Maragnano

Intelligenza Artificiale per la Scienza

Abstract: In this article, we delve into the role of Artificial Intelligence (AI), with a focus on its influence in science and beyond. In the first section, we provide a clear overview of key AI concepts, exploring paradigms such as machine learning, encompassing traditional approaches, and deep learning, specialized in analyzing complex data. The second part focuses on non-scientific applications of AI, with a particular emphasis on board games. Finally, in the third section, we examine the impact of AI on the scientific realm, from microscopic protein structures to global-scale weather models. This article offers a clear and accessible view of AI's influence, embracing the scientific world and beyond.

Key words: Artificial Intelligence, Deep Learning, Science.

Riassunto: In questo articolo esploriamo il ruolo dell'Intelligenza Artificiale (IA), concentrandoci sul suo impatto nella scienza e oltre. Nella prima sezione, forniamo una panoramica chiara dei concetti chiave dell'IA, esplorando paradigmi come il machine learning, che comprende approcci tradizionali, e il deep learning, specializzato nell'analisi di dati complessi. La seconda parte si concentra sulle applicazioni non scientifiche dell'IA, con un'attenzione particolare ai giochi da tavolo. Infine, nella terza sezione, esaminiamo l'impatto dell'IA nel mondo scientifico, dalle strutture microscopiche delle proteine ai modelli meteorologici globali. Questo articolo offre una visione chiara e accessibile dell'influenza dell'IA, abbracciando il mondo scientifico e oltre.

Parole chiave: Intelligenza Artificiale, Deep Learning, Scienza.

Introduzione

L'Intelligenza Artificiale (IA) è una disciplina che negli ultimi anni ha avuto grande risonanza grazie agli ottimi risultati che ha ottenuto. Nel corso degli anni, sempre più modelli di IA sono stati resi pubblici (*open source*), gratis, e facilmente utilizzabili anche da utenti non esperti, contribuendo non solo

Diego Maragnano, Università di Pavia, Dipartimento di Fisica.

ad una rapida diffusione di questi strumenti, ma anche alla produzione di ulteriori modelli e algoritmi da parte di altri sviluppatori e sviluppatrici per affrontare nuovi argomenti e far progredire la disciplina.

In questo senso, è inevitabile pensare al rilascio di ChatGPT da parte di OpenAI, o ai vari modelli di generazione di immagine partendo da frasi testuali come Midjourney.

I metodi di IA sono stati largamente utilizzati, e lo sono sempre più, anche in ambito scientifico, ottenendo a volte risultati strabilianti.

Nelle prossime pagine vorrei accompagnare i lettori e le lettrici verso gli utilizzi dell'IA nella scienza, prima attraverso una panoramica di questa disciplina ampia e diversificata, poi focalizzandoci su alcuni problemi scientifici di particolare rilevanza non solo per il futuro della scienza, ma di tutti noi.

IA: uno sguardo d'insieme

Che cos'è l'IA?

Cominciamo da dove penso sia opportuno cominciare, da una definizione di intelligenza artificiale. Come anticipato in apertura, l'IA è una disciplina che si occupa dello studio teorico e dell'implementazione di sistemi informatici capaci di svolgere compiti che normalmente richiederebbero l'intelligenza umana.

Questa definizione non è né univoca, né universalmente accettata, ma ci permette di cogliere gli elementi distintivi che caratterizzano lo studio dell'IA.

Prima di tutto, l'IA è una disciplina, quindi possiamo considerarla come un termine-ombrello che comprende diverse sotto-discipline, ciascuna con le proprie finalità e metodi.

Compare subito uno stretto legame con l'informatica, sia per lo studio teorico, ad esempio della complessità degli algoritmi, sia per l'implementazione pratica, che può avvenire tramite codici scritti in un opportuno linguaggio di programmazione (detti anche *script*), oppure con sistemi più o meno meccanici che svolgono precisi compiti, come i sistemi robotici.

Infine, il rapporto con l'intelligenza umana. Si tratta di una questione aperta e molto dibattuta: cosa significa intelligenza ed essere intelligenti? Saper fare qualcosa in più di una macchina? Disporre di un'unità di calcolo, nel nostro caso il cervello, più complicata? Come quantifico l'intelligenza di qualcuno o di qualcosa?...Sorgono moltissime domande e moltissimi dubbi, le cui possibili risposte esulano dagli scopi di queste note. Fior fior di ricercatori e ricercatrici si occupano della questione, e lascio a loro il compito di discuterne nelle sedi e nelle modalità più opportune.

Abbiamo iniziato queste note dicendo che oggi l'IA sta avendo molto successo e ha guadagnato molta popolarità. Ma da quanto tempo l'IA stava aspettando questa sua "nuova estate"? E noi, da quanto tempo ci stiamo occupando di IA?

Un po' di storia chiarirà le idee.

Storia dell'IA[1]

L'idea di un'intelligenza artificiale ha le sue radici nella produzione letteraria e cinematografica della prima metà del Novecento. Nel 1927 venne proiettato per la prima volta il film "*Metropolis*", per la regia di Fritz Lang, considerato il paradigma della filmografia fantascientifica successiva. Nel 1950, nel pieno dell'epoca d'oro della letteratura di fantascienza, Isaac Asimov pubblicò "*Io, robot*".

Nello stesso anno Alan Turing, considerato il padre dell'informatica moderna, pubblicò un articolo dal titolo "*Computing Machinery and Intelligence*" (Turing, 1950), in cui discusse un test per valutare la capacità di una macchina di comportarsi in modo intelligente tanto quanto, o almeno in modo indistinguibile da, un umano. In questo articolo Turing chiamò questo test con il nome di "imitation game", ma nel seguito venne (e viene tuttora) chiamato *Test di Turing* dalla comunità. L'interpretazione standard del test di Turing è basata sull'interazione tra tre giocatori, A, B, e C, separati tra di loro. Il giocatore C deve provare a capire chi tra A e B è un computer e chi un essere umano, usando solamente risposte testuali a domande testuali.

Turing tuttavia non mise mai in pratica questo test, per alcune ragioni fondamentali. I computer dell'epoca non erano ancora programmabili nel senso odierno: potevano eseguire comandi, ma non potevano conservare in memoria i risultati per usarli nel seguito. Inoltre, l'utilizzo dei computer era ancora molto costoso, e solamente centri di ricerca ed università potevano permettersi la gestione ed il mantenimento di tali sistemi.

Nel 1955 lo scienziato sociale H. A. Simon, insieme ai ricercatori del RAND (Research ANd Development) A. Newell e J. C. Shaw, pubblicarono "*Logic Theorist*", il primo programma di IA, scritto in IPL (Information Processing Language). "Logic Theorist" venne pensato per dimostrare i teoremi matematici contenuti nell'opera "*Principia Mathematica*" di Whitehead e Russell. In particolare, Logic Theorist riuscì a dimostrare 38 di 52 teoremi del

[1] Per raccontare la storia dell'IA, si dovrebbe anche raccontare la storia dei calcolatori che l'hanno accompagnata e ne hanno permesso lo sviluppo. Per alcune fonti riguardo questi argomenti, rimando agli elementi [a] e [b] della sitografia.

secondo capitolo dell'opera, fornendo dimostrazioni giudicate più dettagliate perfino di quelle fornite dagli stessi autori.

L'occasione per mostrare al mondo i risultati di questo primo ed importante lavoro fu il Dartmouth Summer Research Project on Artificial Intelligence (DSRPAI), organizzato nel 1956 da John McCarthy, Marvin L. Minsky, Nathaniel Rochester, e Claude Shannon, con lo scopo di riunire le diverse correnti di studio che si occupavano di "macchine pensanti". Questo evento fu la prima occasione, almeno tra quelle ufficiali, in cui venne usato il termine intelligenza artificiale, ed è considerato il momento fondante dell'IA come disciplina.

Con la nascita ufficiale dell'IA, iniziò un periodo di fervido lavoro e pionieristici risultati.

Nel 1957 Newell, Simon, e Shaw pubblicarono il programma "*General Problem Solver*", con lo scopo di simulare le capacità umane di risoluzione dei problemi.

Nel 1966 Weizenbaum pubblicò "*ELIZA*", il primo esempio di chatbot (abbreviazione di chatterbot), ideato per esplorare le possibilità di comunicazione tra esseri umani e macchine, la cui versione più famosa tentava di simulare un dialogo con una psicoterapeuta rogersiano. Si racconta che molte persone, tra cui la segretaria dello stesso Weizenbaum, attribuirono sentimenti ed emozioni umane ad ELIZA quando lo utilizzarono. Da questa esperienza nacque poi il cosiddetto effetto ELIZA, che consiste nell'attribuire tratti tipicamente umani a programmi con un'interfaccia testuale. Oggi si può ancora chattare con ELIZA, cercando ad esempio "ELIZA a chatbot therapist" su un motore di ricerca.

Nel 1973 l'Università di Waseda rese pubblico Wabot-1, il primo robot antropomorfo capace di vedere a diverse distanze e in diverse direzioni, conversare in giapponese, camminare, ed afferrare oggetti.

Questi primi sviluppi portarono il DARPA (Defense Advanced Research Project Agency) a finanziare la ricerca sull'IA presso diverse istituzioni. L'ottimismo cresceva, le aspettative erano alte, ma la montagna di ostacoli era ancora più alta, in particolare a causa delle scarse capacità computazionali dei calcolatori dell'epoca.

Tutto ciò rallentò la ricerca nel campo dell'IA, a tal punto che il periodo tra gli anni 1974 e 1980 viene chiamato il primo inverno dell'IA.

Non tutto era perduto. Nel 1980 infatti tornò in auge lo studio dei *sistemi esperti*, di cui si era occupato Edward Feigenbaum negli anni precedenti. Questi sistemi, dopo essere stati addestrati da un esperto umano, possono imitarne il processo decisionale, e fungere così da consulenti per le persone

non esperte. Le intenzioni erano queste, ma ancora una volta le aspettative sull'argomento correvano più velocemente della maturazione tecnologica degli strumenti a disposizione, fino a quando nel 1987 il DARPA decise di interrompere i finanziamenti ai progetti di IA e concentrarsi su progetti con prospettive più solide nel breve termine.

Iniziò così il secondo inverno dell'IA, che durò dal 1987 al 1997.

Nel 1997 però l'IA riuscì a superare un essere umano in un'attività per cui, almeno fino a quel momento, era ritenuta necessaria l'intelligenza umana: il gioco degli scacchi. Parliamo della partita tra Garry Kasparov e DeepBlue, il supercomputer di IBM programmato per giocare a scacchi. Non fu una vittoria assoluta, perché ci fu un primo match nel 1996 vinto da Kasparov, e un secondo match, quello del 1997, vinto da DeepBlue. Ciononostante, l'evento ebbe una notevole risonanza, ridestando così l'interesse generale nei confronti dell'IA.

Nei primi anni del nuovo millennio, diversi eventi segnarono una sempre maggiore presenza dell'IA nella nostra vita quotidiana.

Nel 2002 venne commercializzata la prima generazione di Roomba, il robot che pulisce il pavimento della casa. La capacità di memorizzare il perimetro della casa, cambiare direzione a seconda degli ostacoli che incontra, e tornare alla postazione di ricarica, sono abilità che rendono a tutti gli effetti il robottino un esempio di IA.

Nel 2005 il team di Stanford vinse la seconda edizione del DARPA Grand Challenge, in cui veicoli autonomi di diverse squadre gareggiarono per arrivare alla fine di un percorso prestabilito. In questa edizione, solo 5 dei 23 veicoli partecipanti portarono a termine il percorso. Sempre in tema di competizione con umani, nel 2011 il sistema Watson, sviluppato da IBM, vinse giocando al quiz televisivo "Jeopardy!".

Furono lanciati sul mercato i primi assistenti vocali e di ricerca: Apple introdusse Siri per l'iPhone 4S nel 2011, Google rilasciò il servizio "Google now" nel 2012, e Amazon mise in vendita il primo modello di Amazon Echo nel 2014.

Avvicinandoci sempre più ai giorni nostri, iniziarono ad emergere anche i primi risultati importanti di IA generativa. Nel 2015 venne pubblicato il paper per il Neural Style Transfer, con cui si può trasferire lo stile di un dipinto ad un'altra immagine, ad esempio una foto scattata da noi, e venne rilasciato il modello text-to-image AlignDRAW, che permette di generare immagini partendo da input testuali.

Il laboratorio di ricerca OpenAI, tra i cui fondatori vi è Elon Musk, rilasciò nel 2021 DALL•E, un altro modello per generare immagini da testo scritto.

La stessa OpenAI nel 2022 ha aperto al pubblico l'uso di ChatGPT, che ha definitivamente rilanciato l'IA iniziando una nuova estate della sua storia.

Sempre nel 2022 DeepMind, società controllata di Google, ha rilasciato Gato, un esempio di IA generalista, capace cioè di risolvere diversi compiti, tra cui giocare a videogiochi in stile Atari, muovere dei mattoncini, comporre didascalie per immagini, a seconda del contesto, ossia dell'input che riceve.

Il recente sviluppo dell'IA è dovuto anche alla reciproca interazione tra diversi fattori di natura prettamente tecnologica che caratterizzano la nostra epoca. Oggi abbiamo a disposizione calcolatori molto più potenti e ad un costo irrisorio, se paragonati ai primi computer, grazie alla sempre maggiore possibilità di miniaturizzazione, e conseguente densità di componenti su chip (legge di Moore), e allo sviluppo di unità di elaborazione come le GPU (Graphics Processing Unit). Inoltre, sono stati sviluppati nuovi modelli di IA, come i transformer su cui si basa ChatGPT, e sono disponibili immense quantità di dati, i *big data*, per poter allenare questi modelli.

IA, Machine Learning, e Deep Learning

Quando si parla di IA spesso si sente anche parlare di machine learning e deep learning, a volte usando questi tre termini in modo intercambiabile. Tuttavia, ciò è sbagliato, in quanto il machine learning è un sottoinsieme dell'IA, e il deep learning è a sua volta un sottoinsieme del machine learning.

Procediamo un passo alla volta, cominciando dal machine learning.

Il Machine Learning (ML) si occupa di fornire ad un computer la capacità di imparare senza essere esplicitamente programmato per farlo. In alternativa, possiamo dire che anziché fornire noi le regole al programma, è il programma che trova le regole per noi.

Il Deep Learning (DL) è un sottoinsieme del ML basato sull'uso di reti neurali artificiali, o artificial neural networks. Nel seguito, le chiameremo solamente reti neurali, o neural networks (NN).

Nel dare queste prime definizione, abbiamo usato diverse volte la parola *learning*, apprendimento, senza però darne una definizione precisa per questo contesto.

Vista l'importanza che riveste, è bene approfondire questo concetto.

Scenari di apprendimento

Un sistema di apprendimento è composto da diverse parti. La componente principale è il *modello*, che descrive la relazione tra un input x e un output y. Questa relazione tra input e output è di solito codificata in una funzione parametrica, che dipende cioè da uno o più parametri:

$$h_\theta(x) \rightarrow y$$

Il processo di ***apprendimento***, detto anche ***training*** o ***allenamento***, consiste nel trovare i valori dei parametri che meglio rappresentano la relazione tra input e output analizzando un opportuno set di dati, detto *training set*, attraverso un algoritmo di apprendimento. Solitamente, l'apprendimento avviene minimizzando la differenza tra il comportamento osservato del modello e il comportamento che vorremmo riprodurre. Questa differenza è quantificata da una *loss function*, o *funzione di perdita*. L'obiettivo del processo di apprendimento si traduce nel trovare i valori ottimali dei parametri, ossia i valori che minimizzano la differenza tra i due comportamenti. A tale scopo, vengono solitamente impiegati opportuni algoritmi di ottimizzazione. Una volta che l'algoritmo è allenato, e abbiamo quindi trovato i migliori valori dei parametri del nostro modello, possiamo usarlo per fare previsioni relative al compito per cui è stato addestrato.

Un altro concetto importante in questo contesto è quello di ***generalizzazione***. Vogliamo infatti che il nostro modello fornisca previsioni corrette riguardo a dati che non ha mai visto durante la fase di allenamento. Senza generalizzazione, l'apprendimento sarebbe solo un modo molto complicato di memorizzare dati.

Esistono diverse modalità di apprendimento, basate sulla diversa natura dei ***dati***, o dataset, che abbiamo a disposizione.

Cominciamo chiarendo la tipica struttura di un dataset. Un dataset è composto da un certo numero di *campioni*, ciascuno dei quali è formato da input x e output y.

L'input x contiene la/e grandezza/e che vogliamo utilizzare per predire l'output, e possono essere dati della natura più varia, dai risultati della misurazione di una certa grandezza fisica, ad immagini con una data risoluzione. Gli elementi che compongono l'input, ossia le grandezze che vogliamo usare per predire l'output, sono chiamate *feature*. Il numero di feature dipende dalla natura dell'input. Per esempio, se l'input è formato da immagini in bianco e nero, allora ciascuna feature sarà il valore all'interno di ciascun pixel, e il numero di feature sarà il numero di pixel dell'immagine.

L'output y contiene la/e variabile/i che vogliamo predire, e può essere discreto (0 o 1, 'cane' o 'gatto', 'pioggia' o 'sereno'), oppure continuo (valori della temperatura in una certa regione geografica, i prezzi di un certo asset in un anno).

Se è discreta, allora stiamo svolgendo un compito di ***classificazione***: l'o-

biettivo del modello è quello di predire correttamente la classe del campione che gli stiamo fornendo, partendo dall'input x. Un paio di esempi pratici: l'input per l'algoritmo è un'immagine, e vogliamo che ci dica se è raffigurato un cane o un gatto; diamo in input i dati di temperature, umidità, intensità e direzione del vento in una certa zona geografica in un dato momento, e vogliamo che ci dica se pioverà o sarà sereno nella stessa zona geografica, ma in un momento futuro.

Se è continua, allora stiamo svolgendo un compito di ***regressione***: l'obiettivo non è più classificare correttamente, ma predire uno o più valori, a seconda del problema, il più possibile vicini a quelli corretti. Anche qui un esempio: forniamo all'algoritmo i prezzi giornalieri di un asset negli ultimi sei mesi, e chiediamo che ci dica quale sarà il prezzo nel giorno seguente.

Se i dati sono etichettati, o *labelled*, parleremo di ***apprendimento supervisionato***, o ***supervised learning***. Per usare uno degli esempi precedenti, ciascuna immagine sarà accompagnata dall'etichetta che indica se è raffigurato un cane o un gatto.

Se i dati non sono etichettati, o *unlabelled*, parleremo di ***apprendimento non supervisionato***, o ***unsupervised learning***. In questo caso, avremo le immagini degli animali, ma non saranno accompagnate dall'etichetta che specifica di quale animale si tratta.

Infine, i dati possono essere in parte etichettati e in parte no. In questo caso parleremo di ***apprendimento semi-supervisionato***, o ***semi-supervised learning***.

Per capire meglio la differenza tra dati etichettati e non, consideriamo un dataset con una struttura semplice, ma sufficiente per comprendere la differenza tra dati etichettati e non.

Noto come dataset Iris [c], contiene i dati sul fiore Iris raccolti nel 1936 da Edgar Anderson, poi utilizzati da Fisher per sviluppare un modello di analisi discriminante lineare. Il dataset è composto di 150 righe, e 5 colonne. Ogni riga rappresenta un campione, ossia un esemplare diverso del fiore Iris. Le prime quattro colonne, che sono le feature del campione e saranno l'input di un eventuale modello di apprendimento, contengono lunghezza e larghezza del petalo e del sepalo di un fiore. La quinta colonna, output del modello, può contenere la dicitura "virginica", "setosa", o "versicolor": queste sono tre possibili varietà del fiore Iris. Il dataset contiene 50 campioni per ciascuna di queste tre classi.

Poiché l'output è una variabile discreta, il compito di apprendimento in questo caso sarà di classificazione. Se scegliamo di usare tutte e 5 le colonne

a nostra disposizione, sarà apprendimento supervisionato (ad ogni campione di fiore associamo un'etichetta che ne indica la varietà). Se invece usiamo solo le prime 4 colonne, eliminando l'ultima che contiene l'etichetta della varietà, allora sarà apprendimento non-supervisionato.

Esiste infine un altro approccio, chiamato ***apprendimento per rinforzo***, o ***reinforcement learning***, il cui paradigma è basato sull'interazione tra un agente ed un ambiente.

Prima di capire meglio come affrontare i diversi scenari di apprendimento, esaminiamo la struttura alla base del deep learning, che ci porterà a capire la ragione dietro all'aggettivo "deep" che compare nel nome.

Deep Learning

Nell'ambito del deep learning i diversi compiti di apprendimento vengono svolti usando le reti neurali (artificiali), composte di neuroni disposti secondo un opportuno criterio.

Cominciamo descrivendo la struttura di un neurone artificiale, detto anche percettrone. Supponiamo per semplicità che l'input sia composto di un solo campione, con n feature $[x_1,...,x_n]$. Ciascuna feature viene moltiplicata per un diverso coefficiente $[w_1,...,w_n]$, detto *peso*. Si somma una costante b, detta *bias*. Infine, per ottenere l'output del neurone, si applica una funzione non-lineare, detta *funzione di attivazione*, $a(\,)$. Ciò si traduce in formule:

$$z = b + \sum_{j=1}^{n} x_j w_j \rightarrow y = a(z).$$

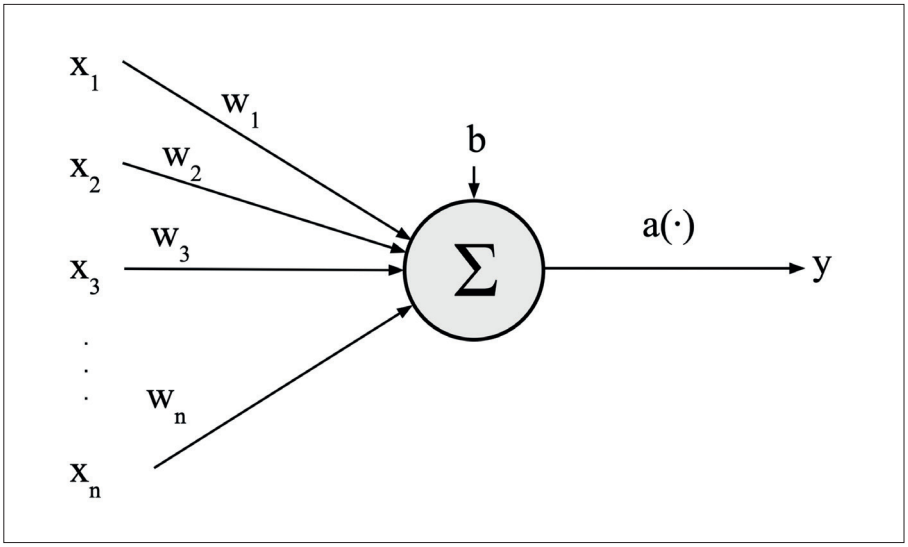
Nel caso di apprendimento supervisionato, l'allenamento di un percettrone consiste quindi nel trovare i migliori valori dei parametri w che permettono al percettrone di predire correttamente l'output y. Una rappresentazione grafica del percettrone è riportata in Figura 1.

A questo punto siamo pronti per costruire una rete neurale.

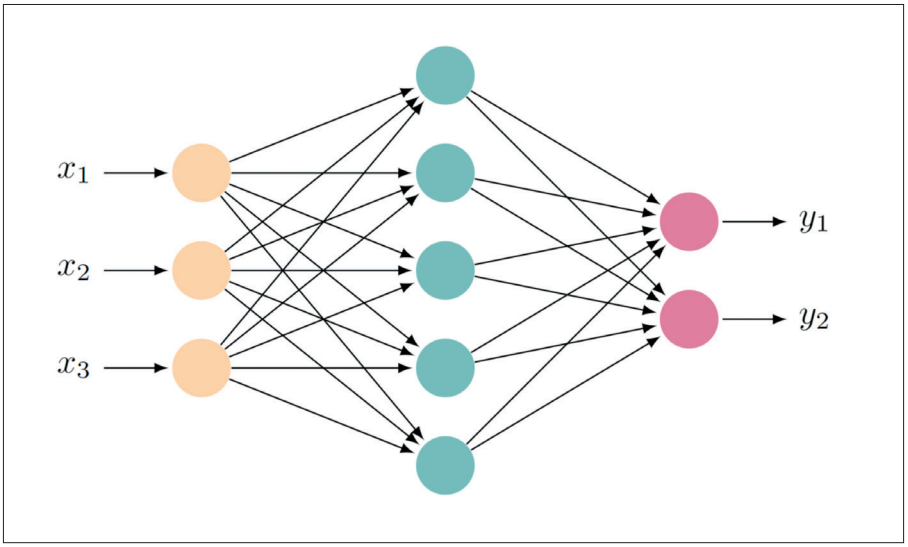
Una rete neurale è composta di *strati*, o layer, di neuroni, ossia di insiemi di neuroni.

Il primo strato si chiama strato di input, o *input layer*, e contiene tanti neuroni quante sono le feature di ciascun campione del nostro dataset. Nel caso del dataset Iris ad esempio, lo strato di input è formato da 4 neuroni.

L'ultimo strato si chiama strato di output, o *output layer*, e contiene tanti neuroni quante sono le classi per un problema di classificazione, o quanti



1. Schema del neurone artificiale, o percettrone.



2. Rappresentazione grafica di un MLP con tre neuroni di input, uno strato nascosto con 5 neuroni, e due neuroni di output.

sono i valori da predire per un problema di regressione. Considerando ancora il dataset Iris, lo strato di output è composto da 3 neuroni.

Tra questi due strati se ne possono inserire altri chiamati strati nascosti, o *hidden layer*, il cui numero di neuroni è stabilito da chi progetta la rete neurale. Determinare il numero di strati nascosti, e il numero di neuroni di ogni strato, è un problema non triviale, la cui soluzione dipende dallo specifico compito in esame.

Se tutti i neuroni di uno strato sono connessi a tutti i neuroni dello strato precedente, e se questo succede per ogni strato (ad eccezione di quello di input, che non ha strati precedenti), allora la rete neurale è anche chiamata Multi-Layer Perceptron, abbreviato in MLP.

Una rappresentazione grafica di un MLP si trova in Figura 2.

Il meccanismo con cui si produce l'output è analogo a quello del percettrone, con la differenza che in una rete neurale l'output di uno strato diventa l'input dello strato successivo. Le formule di prima si riscrivono come segue:

$$z_j^{(l)} = b_j^{(l)} + \sum_{r=0}^{n_l-1} W_{jr}^{(l)}\, x_r^{(l-1)},$$
$$x_j^{(l)} = a\left(z_j^{(l)}\right).$$

dove l è un indice che va da 0 a n_l, e n_l è il numero di strati.

Dopo aver passato in rassegna i concetti essenziali alla base del machine learning e del deep learning, diamo uno sguardo ad alcune delle tecniche che vengono utilizzate nei diversi scenari di apprendimento.

Apprendimento supervisionato

L'apprendimento supervisionato è il tipo di apprendimento che è stato maggiormente studiato finora. Al suo interno si trovano una miriade di modelli diversi, da quelli "classici" di machine learning come la regressione logistica, le (kernel) support vector machine, o le random forest, a quelli di deep learning più avanzati, ottenuti a partire dalla rete neurale "vanilla" che abbiamo discusso nel paragrafo precedente apportando opportune modifiche per renderla performante nei contesti più diversi. Trattandosi di apprendimento supervisionato, fanno tutti uso di dati etichettati.

Apprendimento non-supervisionato

Se i dati non sono etichettati, sono necessarie strategie diverse. Infatti, non possiamo confrontare il risultato del modello con quello esatto contenuto nei

dati, perché i dati non sono etichettati e non contengono alcuna indicazione di quale sia il risultato esatto. Tuttavia, abbiamo diverse alternative disponibili.

Tra le più note e utilizzate ci sono gli algoritmi di *clustering*, che cercano dati simili tra loro nel dataset e li raggruppano; tecniche di *anomaly detection*, o rilevazione delle anomalie, che vanno alla ricerca di campioni inusuali, chiamati anche outlier, nel dataset: ad esempio, se tutte le transazioni della vostra carta di credito riguardano piccole cifre ed avvengono sempre in una zona geografica circoscritta, e all'improvviso iniziano a comparire transazioni di grosse somme dall'altra parte del mondo, queste verrebbero rilevate subito da un algoritmo di anomaly detection.

Una possibile applicazione di apprendimento non-supervisionato è quella dei siti di e-commerce, che consigliano prodotti o servizi agli utenti dopo che questi hanno effettuato un acquisto. Questi suggerimenti potrebbero essere prodotti da modelli allenati con tecniche di *associazione*, in cui l'algoritmo usa pochi attributi chiave e cerca di prevederne altri comunemente associati ai primi: ad esempio, se comprate uno smartphone, è probabile che vi servano anche una cover e una pellicola protettiva per lo schermo, che potrebbe venirvi puntualmente proposta; oppure tecniche di *collaborative filtering*, con cui il modello cerca di indovinare quali prodotti e servizi potrebbero interessarvi, basandosi sul comportamento che avete avuto in passato: ad esempio, se tutte le volte che avete iniziato una nuova saga letteraria ne comprate i primi tre libri, la prossima volta che metterete nel carrello il primo libro di una saga vi verranno consigliati anche i due successivi.

Un altro modello che ha dimostrato la sua efficacia è l'*autoencoder*, che prende i dati di input, ne crea una rappresentazione compressa, e poi cerca di ricreare l'input partendo dalla versione compressa. Un po' come se prendesse un libro, ne facesse un riassunto, e tentasse poi di ricostruire l'intero libro partendo dal riassunto. Un trucco che viene spesso utilizzato con questo modello è quello di fornire una versione "pulita" e una versione "rumorosa" dei dati, così che l'algoritmo impari anche a rimuovere il rumore dai dati.

Allenamento semi-supervisionato

Nel caso di allenamento semi-supervisionato, il dataset contiene sia dati etichettati che non-etichettati. Si usa questo tipo di allenamento quanto estrarre le feature dei dati è particolarmente difficile, oppure etichettare l'intero dataset richiede troppo tempo.

Un modello molto popolare in questo ambito è quello della *Generative Adversarial Network*, o *GAN*. In questo modello c'è un generatore (un modello di apprendimento, che dunque può imparare, ossia un modello i cui

parametri possono essere ottimizzati) che genera dei dati finti, e un dataset contenente dati veri e corretti. Entrambi i tipi di dati vengono inviati ad un classificatore, che deve distinguere quali siano i dati veri e quali quelli finti. Una volta che il classificatore ha fornito il suo responso, questo viene inviato come feedback sia al generatore, affinché impari a generare immagini più simili a quelle vere, sia al classificatore, affinché migliori nel distinguere le immagini.

Per fare un esempio concreto (forse ai limiti della liceità), possiamo pensare il generatore come un falsario d'arte, che cerca di creare dei quadri il più simile possibile a quelli veri contenuti nel dataset, e il classificatore come un esperto d'arte che deve saper riconoscere quando un'opera d'arte è autentica e quando invece è un falso.[2]

Apprendimento per rinforzo

Esaminiamo infine l'ultimo scenario, quello di apprendimento per rinforzo.

Come abbiamo anticipato, l'apprendimento per rinforzo, o reinforcement learning, prevede che un *agente* (l'IA nel nostro caso) impari da un *ambiente* interagendo con esso, attraverso una serie di prove ed errori, e ricevendo delle *ricompense*, negative o positive, come unico feedback in conseguenza delle azioni compiute.

I termini agente, ambiente, e ricompense sono fondamentali nel contesto dell'apprendimento per rinforzo, e per capirne meglio il significato possiamo considerare l'esempio di una persona che inizi a giocare ad un nuovo videogioco. In quanto autore di queste note e nostalgico videogiocatore, scelgo il classico *Super Mario Bros.*, pubblicato per la prima volta nel 1985 dalla Nintendo per il NES. Guardiamo il primo frame, riportato in Figura 3.

Nel linguaggio dell'apprendimento per rinforzo, il videogiocatore (o equivalentemente Mario) è l'agente, mentre il mondo di gioco è l'ambiente.

Quando il gioco inizia, l'ambiente (il videogioco) fornisce all'agente (il videogiocatore) un primo frame di gioco, detto *stato*, che chiamiamo S_0. Uno stato è una descrizione, completa o parziale, dell'ambiente di gioco. In questo caso, lo stato ci dice che Mario si trova in una precisa posizione iniziale, su un terreno al di sotto del quale non può andare, con diversi blocchi sospesi a mezz'aria, e un nemico di fronte a lui.

[2] Se pensiamo alla vicenda delle "teste di Modigliani" del 1984 come un'istanza di una GAN, allora il generatore (più d'uno in questo caso) fu molto bravo nel creare i dati fittizi, ingannando diversi classificatori esperti chiamati in causa per capire se le sculture fossero davvero opera di Modigliani.

3. Primo frame di Super Mario Bros., Nintendo, 1985. [Credits: https://huggingface.co/learn/deep-rl-course/unit1/rl-framework]

A questo punto l'agente valuta lo stato, e sceglie un'azione A_0 da compiere tra quelle ammesse dal videogioco: si muove verso destra.

L'azione scelta porterà l'agente in nuovo stato, ossia un nuovo frame del videogioco, chiamato S_1.

A questo punto l'ambiente fornisce all'agente una ricompensa R_1, diversa a seconda dello stato in cui si è venuto a trovare l'agente. Se Mario non è incappato in alcun nemico, e dunque può continuare a giocare, la ricompensa sarà positiva. Se invece deve ricominciare il livello perché ha perso contro un nemico, allora la ricompensa sarà negativa.

Questa successione di passaggi, che è alla base dell'apprendimento per rinforzo, viene poi ripetuta: in base allo stato S_1, l'agente compie un'azione A_1 che lo porta in un nuovo stato S_2, accompagnato da una ricompensa R_2 e così via.

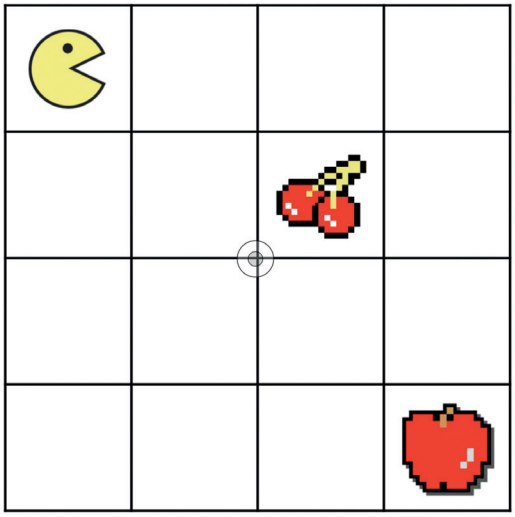
Affinché l'apprendimento abbia luogo e progredisca, dobbiamo trovare un obiettivo di apprendimento, che ci informi su quanto il nostro agente stia effettivamente imparando, e come raggiungere tale obiettivo.

Nell'apprendimento supervisionato l'obiettivo era minimizzare la differenza tra il comportamento atteso e quello osservato. Nell'apprendimento per rinforzo invece l'obiettivo è massimizzare la *ricompensa cumulativa*, chiamata anche *ritorno atteso*, somma delle ricompense che l'agente riceve dopo ogni azione. In questo senso, dobbiamo trovare un modo di far compiere all'agente la sequenza di azioni che massimizza la ricompensa cumulativa.

Per scegliere in modo opportuno quale azione compiere in base allo stato in cui si trova, l'agente dovrà usare il cervello (ovviamente), che nel linguaggio dell'apprendimento per rinforzo è rappresentato da una *policy*. Una policy, che indichiamo con la lettera π, definisce il comportamento dell'agente nell'ambiente, e comprende le azioni che l'agente dovrebbe compiere per ogni possibile stato.

L'obiettivo dell'apprendimento per rinforzo si traduce quindi nel trovare quella policy che massimizza la ricompensa cumulativa, quando l'agente agisce secondo quanto indicato da quella policy.

Per capire meglio come funzionino le policy, guardiamo alla situazione riportata di seguito.



L'agente in questo caso è Pac-Man, che può muoversi all'interno di una griglia con l'obiettivo di mangiare della frutta.

Lo stato dell'agente è rappresentato dalla sua posizione sulla griglia: $S = (x,y)$. Lo stato iniziale è dunque $S_0 = (1,1)$.

Le azioni possibili sono, in generale, quattro: su, giù, destra, sinistra.

Le ricompense sono così definite: trovarsi in una casella vuota fornisce una ricompensa -1, mangiare le ciliegie +5, mangiare la mela +10.

Una prima policy π_1 che l'agente può seguire è data da questa sequenza di azioni: {giù, destra, destra}. Se l'agente si comporta come indicato da questa policy, arriverà a mangiare le ciliegie, con una ricompensa complessiva di +3.

Un'altra policy π_2 è la seguente: {destra, destra, destra, giù, giù, giù}. Così

facendo l'agente arriverà a mangiare la mela, ricevendo una ricompensa complessiva di +5.

Da questa prima analisi sembra quindi che all'agente convenga seguire la seconda policy. Eppure non è detto che sia la scelta migliore.

Supponiamo per un momento che l'agente inizi il gioco, ma abbia un numero limitato di mosse da poter compiere, diciamo 30. Decide di utilizzare subito la policy π_2, scoprendo che gli costa 6 mosse e una ricompensa +5.

A questo punto l'agente ha due alternative davanti a sé: sfruttare la conoscenza che ha già, oppure esplorare e cercare nuove strategie.

Se sfrutta la conoscenza che ha già, può ripetere la policy π_2 per 5 volte prima di esaurire le mosse a sua disposizione, con una ricompensa complessiva di + 25.

Se invece decide di esplorare, provando policy alternative, potrebbe scoprire la policy π_1, che costa 3 mosse e porta una ricompensa +3. Questa policy può essere ripetuta per 10 volte prima di esaurire le mosse, portando una ricompensa complessiva +30.

Dopo questa ulteriore analisi, la policy migliore sembra essere la prima.

Quello che abbiamo appena analizzato è detto *exploitation-exploration tradeoff*, ossia un tradeoff tra seguire e ripetere una policy conosciuta, oppure esplorarne di nuove sperando di guadagnare in termini di ricompensa cumulativa. Per la natura stessa del tradeoff, una risposta univoca a quale sia la strategia migliore non c'è, ma dipende dallo specifico problema di apprendimento che si sta studiando.

Un ultimo commento prima di procedere. Da quanto abbiamo detto, l'apprendimento per rinforzo si traduce nel trovare della policy ottimale. Questa può essere imparata tramite dei metodi "classici", nei cui dettagli non ci addentreremo, oppure tramite una rete neurale, entrando così nel reame del *deep reinforcement learning*. Con questa rappresentazione, l'obiettivo dell'apprendimento diventa determinare i parametri ottimali della rete neurale, che corrispondono a loro volta alla policy ottimale.

IA e giochi

Tra gli eventi rilevanti della storia dell'IA presentata all'inizio compare la vittoria di DeepBlue contro Kasparov nel 1997. Questa non fu però la prima incursione dell'IA nel mondo dei giochi. Già nel 1950 Shannon discusse la possibilità di programmare un computer per giocare a scacchi (Shannon,

1950). Nel 1959 Arthur Samuel pubblicò un articolo in cui discuteva l'implementazione di un algoritmo di IA per giocare a dama (Samuel, 1959). Gerald Tesauro nel 1995 studiò come l'IA potesse imparare a giocare a backgammon (Tesauro, 1995).

Questi algoritmi venivano realizzati implementando a mano migliaia di possibili strategie e mosse diverse, pensate da giocatori umani esperti, nel tentativo di immaginare tutti gli scenari che l'IA avrebbe potuto dover affrontare durante una partita.

Lo studio di come un modello di IA possa imparare e guadagnare abilità nei vari giochi è proseguito negli anni. Tra gli altri, OpenAI ha pubblicato diversi lavori in cui allena modelli di IA a giocare a Dota 2 (videogioco multiplayer online), wrestling, Minecraft (videogioco sandbox), e anche a nascondino (Berner, 2019 - Baker, 2020).

OpenAI si è prodigata anche nello studio di modelli di IA per i giochi da tavolo, focalizzandosi in particolare sull'apprendimento per rinforzo nei giochi di scacchi, shogi (versione cinese degli scacchi), e go, ottenendo notevoli risultati. Tuttavia, la loro strategia di allenamento del modello non era basata sull'implementazione manuale di strategie di gioco, ma su un approccio chiamato *self play*.

La prima differenza tra l'apprendimento per rinforzo tradizionale e quello adottato per i giochi, è la presenza di più di un agente. Nell'introduzione all'apprendimento per rinforzo abbiamo considerato esempi con un solo agente in campo. Nel contesto dei giochi ci sono almeno due agenti (altrimenti che giochi sarebbero?) che interagiscono in un ambiente comune. L'ambiente in cui si trovano condiziona il comportamento degli agenti. Se si trovano in un ambiente cooperativo, agiranno in modo che tutti abbiano il massimo beneficio, come i robot in un magazzino che scaricano e caricano la merce. Se l'ambiente è competitivo, allora ciascun agente cercherà di massimizzare il proprio beneficio, a scapito di quello degli avversari, come in una partita di tennis. L'ambiente può anche essere misto, con un bilancio tra il beneficio/perdita di un agente e quello degli altri agenti è più complesso, come in una partita di calcio.

In un contesto come quello appena descritto di sistemi multi-agente, il self play è una delle possibili strategie per l'allenamento. L'idea alla base è molto semplice, e prende ispirazione da come noi umani impariamo e miglioriamo quando iniziamo un nuovo gioco. Prima di tutto, servono le regole base, che ci permettono di capire quale sia lo scopo del gioco e quali siano le modalità permesse per raggiungerlo. La situazione ideale sarebbe poi quella di giocare contro un avversario del nostro stesso livello, che diventi più bravo

mano a mano che lo diventiamo anche noi. Infatti, l'avversario non deve essere troppo forte, altrimenti non impariamo affatto, e neanche troppo debole, altrimenti impariamo delle strategie che saranno inutili quando andremo ad affrontare avversari più bravi.

Per implementare questa situazione, si fa giocare l'agente contro delle copie di sé stesso, e una volta che ha imparato qualcosa, si rende l'avversario più forte (ossia si usano delle copie dell'agente che hanno a loro volta imparato qualcosa, scelte tra le copie dell'iterazione precedente).

Come possiamo sapere se l'agente ha imparato qualcosa oppure no? Ossia, come si quantifica l'apprendimento nel contesto del self play? La ricompensa complessiva dell'apprendimento per rinforzo tradizionale non è una buona metrica, in quanto dipende dal livello dell'avversario. Si usa invece il *punteggio di Elo*, o *Elo score*, legato al livello di abilità relativo di due giocatori in un gioco a somma zero (un gioco in cui uno vince e uno perde, e il guadagno di uno è perfettamente bilanciato dalla perdita dell'altro). Il termine "relativo" nella definizione è cruciale, perché ci dice che il punteggio di Elo dipende non solo da quanto sia abile l'avversario, ma anche dall'esito della partita contro di lui.

Proviamo a chiarire con un esempio. Inizia la partita, ed entrambi i giocatori hanno un punteggio di Elo iniziale. Al termine della partita, uno avrà vinto e uno avrà perso. Il vincitore guadagnerà dei punti che andranno ad aumentare il suo punteggio di Elo, mentre lo sconfitto ne perderà. Quanti siano questi punti dipende dalla differenza nel livello dei due giocatori. Se il giocatore più abile vince, allora il suo punteggio di Elo aumenterà di poco, e di altrettanto poco diminuirà quello dello sconfitto: in questo caso ci si aspettava che vincesse, quindi non avrà imparato molto da questa partita. Se il giocatore meno abile vince, allora il suo punteggio di Elo aumenterà di molto, e conseguentemente diminuirà quello del giocatore più abile: questo risultato era inatteso, ed è molto probabile che il giocatore meno abile abbia imparato qualcosa che gli ha permesso di battere l'avversario, quindi il suo punteggio di Elo aumenta di molto. Infine, se finisce in pareggio, tipicamente il punteggio del giocatore meno abile aumenta di pochi punti, e di altrettanto diminuisce quello del più abile.

Nel caso di giochi a squadre, è possibile usare la media dei punteggi di Elo dei singoli giocatori come metrica.

Il self play è alla base di *AlphaZero*, il modello di IA sviluppato da DeepMind capace di giocare a scacchi, shogi e go (Silver, 2017 - Silver, 2018). AlphaZero è un modello di deep reinforcement learning che all'inizio conosce solamente le regole base del gioco, e impara attraverso il self play, giocando

contro sé stesso. Possiamo dire che impara quale sia la mossa più utile per il futuro, data l'attuale configurazione della scacchiera. Ad ogni turno sceglie la successiva mossa da compiere cercando in una piccola frazione delle mosse considerate dagli algoritmi tradizionali. Per esempio, nel caso degli scacchi AlphaZero cerca in uno spazio di 60000 possibili mosse, contro 60 milioni di Stockfish, il migliore algoritmo tradizionale per gli scacchi.

Terminato l'allenamento, AlphaZero viene fatto giocare contro i migliori algoritmi dei tre giochi considerati: Stockfish per gli scacchi, Elmo per gli shogi, e AlphaGoZero per il go, suo predecessore che aveva battuto Lee Sedol nel 2016.

Il tempo necessario per completare l'allenamento e per superare gli avversari (ossia ottenere un punteggio di Elo più alto), dipende dal gioco. Negli scacchi, in 4 ore AlphaZero ha superato Stockfish, e terminato l'allenamento in 9 ore. Negli shogi, dopo 2 ore era più forte di Elmo, e in 12 ore ha terminato l'allenamento. Infine, dopo 30 ore era più bravo del suo predecessore a giocare a go, e ha impiegato 13 giorni a terminare l'allenamento.

Guardando le statistiche, AlphaZero ha vinto il 15.5% delle partite a scacchi (con moltissimi pareggi e pochissime sconfitte nella percentuale rimanente), il 91.2% delle partite a shogi, e il 61.3 % di quelle a shogi.

Se ci rivolgiamo brevemente al mondo dei videogiochi, notiamo che l'uso dell'IA è molto frequente, in particolare per il controllo dei personaggi non-giocanti (non-playable characters, NPCs). Per esempio, un nemico potrebbe comportarsi diversamente a seconda della strategia che adottiamo per affrontarlo o a seconda della progressione nella trama.

Oppure, possiamo considerare il personaggio giocante come un modello di apprendimento, e allenarlo per trovare la strategia migliore per affrontare il gioco. Questo si può fare usando il (deep) reinforcement learning, oppure un algoritmo genetico.

Gli *algoritmi genetici* simulano l'azione della selezione naturale su una popolazione di possibili soluzioni (copie del personaggio) per un problema di ottimizzazione (arrivare alla fine del gioco), attraverso diverse fasi:

1. Inizializzazione: viene creato un insieme di soluzioni casuali.
2. Valutazione: si definisce una funzione di fitness, che quantifica la qualità di una soluzione in relazione al problema considerato.
3. Selezione: vengono scelte alcune soluzioni per passare alla generazione successiva.
4. Ricombinazione: si ricombinano le soluzioni.
5. Mutazione: vengono introdotte opportune modifiche per le nuove soluzioni.
6. Si ripetono i punti da 2 a 6 fino a quando il problema non è risolto.

Si tratta di un approccio molto generale e potente, di cui potete trovare qualche esempio sul canale Youtube di Code Bullet [d], in cui modelli di IA imparano a giocare a "Donkey Kong vs Mario", "Jump King", "Tetris", "Snake", ma anche a camminare, guidare, risolvere cubi di Rubik (a volte molto grandi).

IA per la scienza

Cominciamo ora l'ultima sezione di queste note, in cui finalmente esaminiamo alcune applicazioni di metodi di IA in ambito scientifico. In particolare, ci occuperemo di previsione della struttura delle proteine, controllo magnetico nei tokamak, e previsioni meteorologiche. Le applicazioni scientifiche dell'IA sono innumerevoli, e queste tre rappresentano una mia personale scelta. Il motivo è l'importanza dei possibili risvolti applicativi nella medicina, includendo tutte le discipline ad essa legate, e nella lotta al cambiamento climatico, per lo studio sia di nuove fonti di energia, sia di metodi più efficienti per capire come e quanto il clima cambierà in futuro.

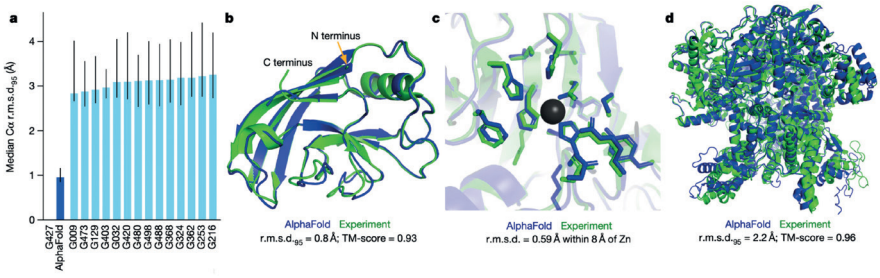
Previsione della struttura delle proteine

Il problema della previsione delle proteine consiste nel determinare la struttura tridimensionale di una proteina partendo dalla sequenza di aminoacidi che la caratterizza. Più precisamente, consiste nel determinare la struttura secondaria, terziaria, ed eventualmente quaternaria, partendo da quella primaria. Si tratta di uno dei più importanti problemi della biologia computazionale, la cui difficoltà principale sta nella complessità delle interazioni che regolano la formazione della struttura tridimensionale della proteina.

Esistono diversi metodi per studiare il problema della previsione della struttura delle proteine, che vengono valutati ogni due anni nel CASP (Critical Assessment of Techniques for Protein Structure Prediction).

Come la struttura di un macchinario di permette di capire cosa faccia, analogamente la struttura di una proteina ci permette di capire quale sia la sua funzione. Prevedere con maggiore accuratezza quale sia la struttura delle proteine avrebbe ricadute positive nello studio contro la resistenza agli antibiotici, l'inquinamento da microplastiche, e indirettamente anche contro il cambiamento climatico.

DeepMind ha affrontato e risolto (almeno parzialmente) questo problema, attraverso lo sviluppo di AlphaFold, che ha portato a due pubblicazioni scientifiche (Jumper, 2021 - Tunyasuvunakool, 2021), il rilascio di un codice open source e di un database con la struttura delle proteine già studiate. In



4. a): Errore compiuto da AlphaFold nella ricostruzione della struttura tridimensionale delle proteine al CASP14. b), c), d): esempi di strutture tridimensionali ricostruite da AlphaFold (blu) e determinate sperimentalmente (verde).

particolare, i ricercatori e le ricercatrici di DeepMind dichiarano di aver studiato quasi tutto il proteoma umano usando AlphaFold, e di poter predire con sufficiente accuratezza la struttura di circa il 58% di tale proteoma. A questo si aggiunge il proteoma di altri 20 organismi biologicamente rilevanti, per un totale di circa 350000 strutture.

Se curiosiamo un po' nei due articoli pubblicati, notiamo subito come l'architettura sia ben più complicata di quelle che sono state presentate all'inizio di queste note. La previsione della struttura delle proteine è un problema molto complicato, ed è stato necessario introdurre delle opportune modifiche ad una o più delle architetture "standard". Tuttavia, se guardiamo da una prospettiva più ampia e senza scendere nei dettagli, il modello accetta come input la struttura primaria della proteina (la sequenza di amminoacidi) e restituisce come output la struttura tridimensionale. In Figura 4 sono riportati alcuni dei risultati ottenuti da AlphaFold nel CASP14. Nella figura 4a vediamo che l'errore commesso nella ricostruzione è molto minore di quello degli altri metodi che sono stati studiati, mentre in 4b, 4c, e 4d vediamo alcuni esempi di strutture ricostruite da AlphaFold (in blu) confrontate con la struttura esatta della proteina, determinata sperimentalmente (in verde).

Diversi studi hanno già beneficiato dei risultati di AlphaFold.

DeepMind sta usando AlphaFold nella collaborazione intrapresa con DnDI (Drugs for Neglected Diseases Initiative) per cercare cure per le malattie che affliggono i paesi in via di sviluppo, malattie del sonno, malattia di Chagas e leishmaniosi umana.

La collaborazione con il Centre for Enzyme Innovation dell'Università di Portsmouth ha l'obiettivo di ingegnerizzare più velocemente gli enzimi per riciclare alcune delle plastiche monouso più inquinanti.

Diverse università hanno utilizzato AlphaFold per confermare la struttura di alcune proteine legate a SARS-CoV-2.

Lo studio è proseguito negli ultimi due anni, e notizie più recenti si possono reperire direttamente dal sito di DeepMind [e].

Controllo magnetico nei tokamak

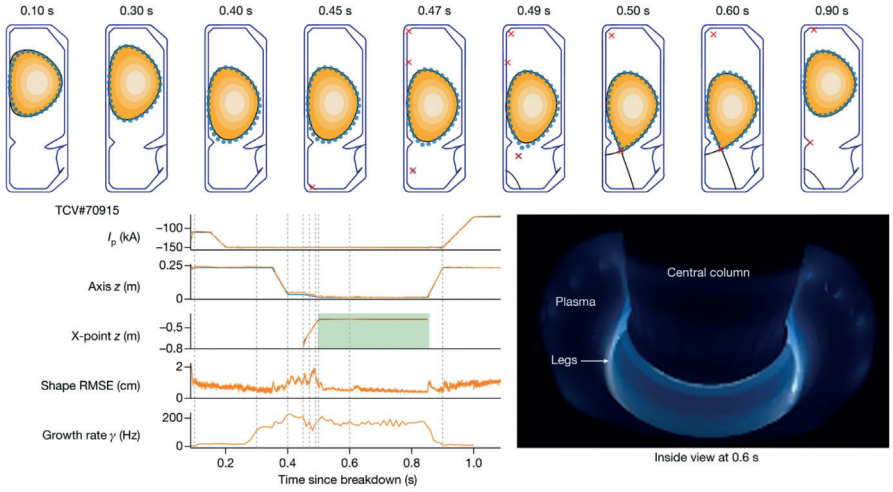
Il termine tokamak deriva da un acronimo russo che significa "camera toroidale con bobine magnetiche", e rappresenta perfettamente quello che il tokamak è: un dispositivo che confina un plasma in una struttura toroidale tramite potenti campi magnetici. I tokamak vengono usati negli studi sulla fusione nucleare, in quanto il plasma, composto di particelle cariche elettricamente e molto poco denso, è l'ambiente ideale affinché la fusione nucleare avvenga.

Al momento si stanno studiando diverse configurazioni e forme della distribuzione del plasma all'interno del tokamak, al fine di ottimizzare la stabilità ed il confinamento dell'energia nell'ambito del progetto ITER, in corso nel sud della Francia per dimostrare la fattibilità della fusione nucleare come fonte di energia su larga scala e carbon-free [f].

Si tratta di un compito molto difficile. Infatti, confinare ciascuna delle configurazioni di plasma all'interno del tokamak richiede la progettazione di un apposito sistema di feedback, che possa agire sul campo magnetico, modificandolo opportunamente, tramite il controllo molto preciso di tensioni e correnti elettriche di diverse bobine. Queste sono a loro volta accoppiate magneticamente con il plasma. Controllando le bobine, l'obiettivo è che all'interno del tokamak il plasma abbia la corrente, la posizione, e la forma desiderate. Il controllo magnetico del tokamak è quindi un problema di controllo non-lineare, tempo-dipendente, e multivariato. Già solo questa sequenza di aggettivi potrebbe essere sufficiente a rendere conto di quanto sia difficile.

L'approccio convenzionale (ridotto ai minimi termini) è il seguente. Si calcolano preventivamente i valori delle tensioni e delle correnti per il controllo delle bobine. Si usano dei sistemi di controllo PID per impostare la posizione del plasma nella direzione verticale e radiale, e la corrente di plasma, senza che interferiscano uno con l'altro. Infine, un sistema di controllo esterno si occupa di selezionare la forma del plasma. Questo sistema è molto dispendioso dal punto di vista della progettazione e dell'attuazione, poiché va opportunamente modificato ogni volta che viene modificata la configurazione del plasma.

Nel 2022 DeepMind e lo Swiss Plasma Center di EPFL hanno pubblicato un articolo in cui proponevano un nuovo approccio al problema del controllo magnetico del tokamak basato sul deep reinforcement learning (Degrave,



5. *In alto:* diverse posizioni e forme del plasma all'interno del tokamak. I cerchi blu rappresentano l'obiettivo, mentre la linea nera continua rappresenta la ricostruzione fatta dopo l'esperimento. *In basso a sinistra*: tracce di diverse quantità come obiettivo (blu) e ricostruite dall'esperimento (arancione). *In basso a destra*: immagine all'interno del tokamak che mostra la configurazione a 0.6 s.

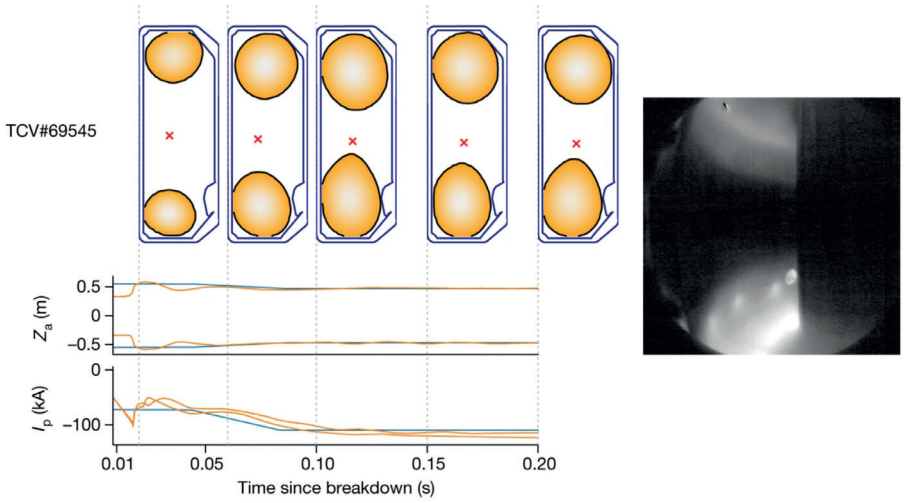
2022). I ricercatori sono riusciti a riformulare il problema all'interno del framework dell'apprendimento per rinforzo attraverso alcuni passaggi.

Prima di tutto, l'obiettivo sperimentale determinato dalle proprietà del plasma, per esempio posizione, corrente, e forma, è stato tradotto in una opportuna funzione di ricompensa da massimizzare.

Una rete neurale interagisce con un simulatore di tokamak per determinare quale sia la policy ottimale.

Infine, la rete neurale (o meglio, la policy ottimale che la rete ha imparato) viene usata per il controllo in tempo reale di un vero tokamak. Più precisamente, l'esperimento sulla macchina reale comincia usando il tradizionale sistema di controllo per mantenere il plasma nella configurazione desiderata, e dopo un lasso di tempo deciso in precedenza, detto tempo di handover, il controllo del plasma viene passato alla rete neurale.

Vediamo ora alcuni risultati di questo nuovo approccio. Nella Figura 5 in alto vediamo come il modello di deep reinforcement learning sia stato particolarmente efficace nel riprodurre e controllare il plasma nelle diverse posizioni e forme desiderate. In particolare quella a 0.6 s è riportata anche nel pannello in basso a destra. Nel pannello in basso a sinistra invece vediamo



6. *A sinistra*: in alto, configurazione a gocce del plasma; in basso, i valori di posizione lungo la direzione verticale e di corrente di plasma attesi (linea blu) e ricostruiti (arancione) per la configurazione a gocce. *A destra*: immagine dall'interno del tokamak in cui le due gocce sono ben visibili.

come il modello sia riuscito a mantenere la proprietà desiderate nel tempo, rimanendo molto vicino a quelli che erano i valori fissati come obiettivo (le linee continue blu nel grafico). Non volendosi imitare alla configurazioni tradizionali, i ricercatori hanno deciso di provare a sperimentare il controllo magnetico del plasma in configurazioni nuove. In Figura 6 vediamo come sia stata realizzata e controllata la configurazione "a gocce", in cui si formano due distinti lobi di plasma, uno nella parte alta del tokamak, e uno nella parte bassa.

Previsioni meteorologiche

Le condizioni meteorologiche della zona geografica in cui viviamo riguardano sono importanti per molti aspetti quotidiani della nostra vita, per esempio per sapere se sia il caso di portare con sé un ombrello quando si esce di casa, e lo sono ancora di più in caso di eventi eccezionali come siccità e temperature molto alte, grandine, e temporali o tempeste di vento particolarmente intense. Poter fare delle previsioni affidabili delle condizioni meteorologiche future, per lo meno a breve termine, è dunque fondamentale, e lo diventerà

sempre più negli anni a venire a causa del cambiamento climatico. Anche per questo problema l'IA può venire in nostro aiuto.

Cominciamo però dal sistema tradizionale per le previsioni meteorologiche, i modelli numerici per le previsioni meteorologiche, o numerical weather prediction models. In questi modelli i diversi stati atmosferici (ossia gli stati dell'atmosfera intesa come un fluido) sono rappresentati come elementi di una griglia discreta, individuata da diversi valori di posizione e altitudine, e vengono risolte numericamente le equazioni differenziali che governano le transizioni tra questi stati.

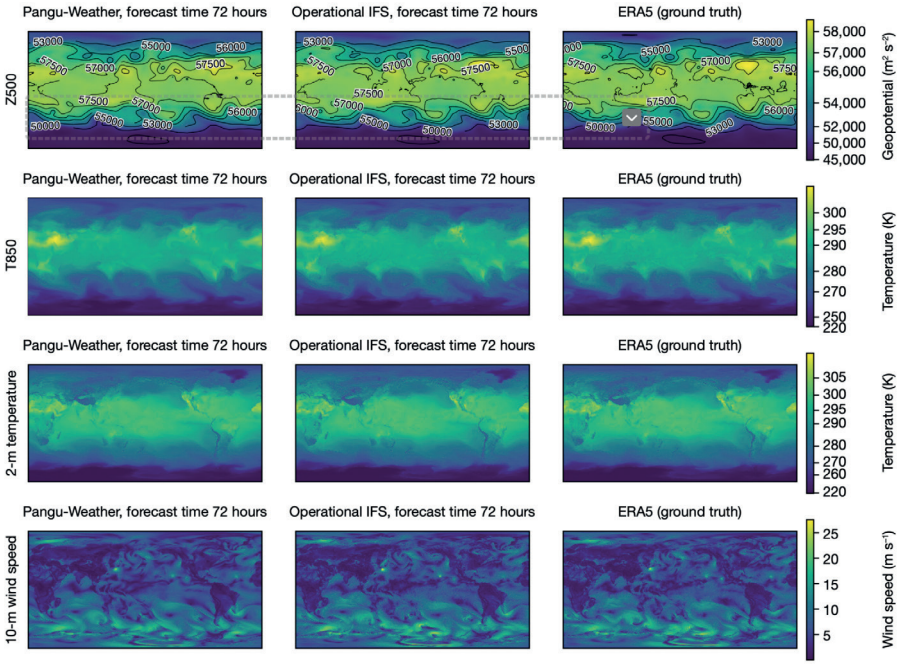
All'inizio di luglio 2023 un gruppo di ricercatori di Huawei Cloud, con sede a Shenzen, in Cina, pubblica un articolo in cui viene presentato Pangu-Weather, un modello di IA basato su un'architettura costruita appositamente per l'atmosfera terrestre per previsioni meteorologiche globali a medio termine (5-10 giorni) (Bi, 2023). Allenato su 39 anni di dati, raccolti tra il 1979 e il 2017, è stato confrontato con il miglior modello numerico per le previsioni dell'ECMWF (European Centre for Medium-range Weather Forecast).

L'input di Pangu-Weather è composto da dati di rianalisi[3] meteorologica in un dato punto (geografico) nel tempo (passato o presente), mentre l'output è composto da dati dello stesso formato, nello stesso punto, ma in un tempo futuro rispetto a quelli di input.

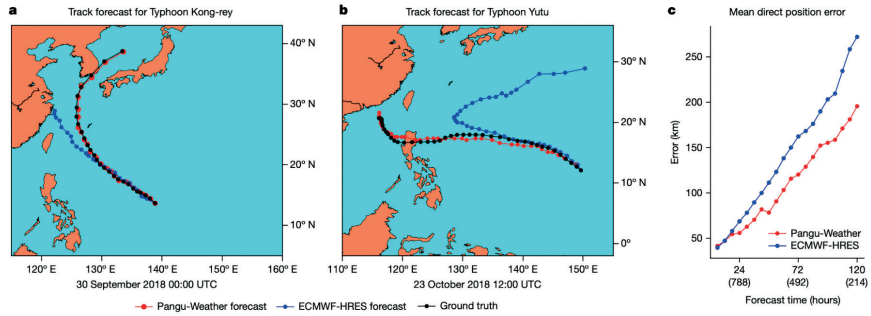
Pangu-Weather è stato testato su ERA5, la quinta generazione dei dati di rianalisi dell'ECMWF, dimostrandosi efficace sia nella previsione di diverse variabili "ordinarie", quali temperatura, velocità del vento, umidità specifica, e pressione, sia di eventi estremi come tornado o cicloni. Inoltre, è stato 10000 volte più veloce dell'IFS (Integrated Forecasting System) dell'ECMWF.

Vediamo alcuni risultati. In Figura 7 viene mostrata la previsione globale, su 3 giorni, di 4 variabili atmosferiche. Si può notare come entrambi i modelli, Pangu-Weather e IFS, siano vicini ai valori misurati da ERA5. Tuttavia, nell'articolo viene fatto notare come Pangu-Weather produca curve di livello più smussate, più "morbide", ad indicare una maggiore probabilità di predire valori simili per regioni geografiche vicine. In Figura 8 vengono riportati i

[3] I dati di rianalisi permettono di avere informazioni sugli stati atmosferici riferite anche a molti anni nel passato. Si comincia con una fase di raccolta dei dati, in cui si combinano tutte le osservazioni disponibili (stazioni a terra, navi, aerei, satelliti). Questi dati vengono poi usati per creare una griglia uniforme che farà da dato iniziale per il modello numerico. Questa operazione, se fatta partendo da dati raccolti nel passato, permette di costruire una lunga sequenza temporale di istantanee di condizioni atmosferiche, che viene detta dataset di rianalisi.



7. Previsione globale, su 3 giorni, di 4 variabili atmosferiche, dall'alto verso il basso: Z500, T850, 2-m temperature, e 10-m wind speed. Da sinistra verso destra: previsione di Pangu-Weather, previsione di IFS, dati di ERA5.



8. Localizzazione di cicloni tropicali ed errore medio in funzione del tempo di previsione. a), b): traiettoria misurata (nero), prevista da Pangu-Weather (rosso), e prevista ds ECMWF-HRES (blu). c) Errore medio commesso da Pangu-Weather (rosso) e da ECMWF-HRES (blu) sulla posizione degli 88 cicloni tropicali del 2019. La media viene calcolata su tutti gli 88 cicloni, e i numeri tra parentesi sotto le etichette dell'asse orizzontale indicano il numero di campioni usati nel calcolo della media per le ore corrispondenti.

risultati di localizzazione di due degli 88 cicloni tropicali del 2019, e viene riportata la media dell'errore nella previsione della posizione, in funzione del tempo per cui si sta facendo la previsione. Pangu-Weather riesce a predire molto bene la posizione dei cicloni, meglio del sistema HRES (High RESolution) dell'ECMWF, commettendo un errore medio più piccolo.

Gli autori non mancano però di evidenziare i limiti del loro modello. L'input è costituito da dati di rianalisi, quindi da una sorta di ricostruzione storica dei dati del passato, mentre nel presente abbiamo a disposizione dati che derivano da misurazioni dirette. Alcune variabili meteorologiche non sono state investigate, ad esempio le precipitazioni. Questo potrebbe portare il modello a non saper riconoscere eventi su piccola scala come quelli che danno inizio ai tornado. Infine, il fatto che Pangu-Weather produca risultati più smussati potrebbe comportare una sottostima della grandezza di eventi meteorologici estremi. In questo lavoro sono stati studiati i cicloni tropicali, ma altro lavoro deve essere fatto per migliorare il modello.

Conclusioni

In queste pagine abbiamo introdotto alcuni elementi essenziali di Intelligenza Artificiale, Machine Learning, Deep Learning, e Reinforcement Learning. Abbiamo poi visto alcune delle importanti applicazioni che l'IA ha avuto nel mondo dei (video)giochi, e in ambito scientifico, studiando la struttura delle proteine, il confinamento magnetico nei tokamak, e le previsioni meteorologiche. I metodi di IA applicati in ambito scientifico permettono di ottenere informazioni importanti sui fenomeni che stiamo studiando, che magari i metodi tradizionali non ci permetterebbero di ottenere altrimenti.

Ci sono molti altri argomenti di cui parlare, a partire dallo studio dell'etica del'IA, di cui è e sarà sempre più necessario occuparsi, data la diffusione ormai capillare dell'IA, che potremmo ormai definire a pieno titolo una tecnologia trasformativa.

Concludo con la fiducia che l'IA possa aiutare a far progredire lo studio e la comprensione dei problemi che più ci riguardano, in particolare il cambiamento climatico e le malattie, per lasciare il mondo un po' migliore di come l'abbiamo trovato.

Bibliografia

Baker B. *et al.*, 2022, *Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos.*, arXiv preprint, https://doi.org/10.48550/arXiv.2206.11795.

Baker B. *et al.*, 2020, *Emergent tool use from multi-agent autocurricula.*, arXiv preprint, https://doi.org/10.48550/arXiv.1909.07528.

Bansal T. *et al.*, 2018, *Emergent complexity via multi-agent competition.*, arXiv preprint, https://doi.org/10.48550/arXiv.1710.03748.

Berner C. *et al.*, 2019, *Dota 2 with large scale deep reinforcement learning.,* arXiv preprint, https://doi.org/10.48550/arXiv.1912.06680.

Bi K. *et al.*, 2023, *Accurate medium-range global weather forecasting with 3D neural networks*, «Nature», 619, 7970, pp. 533, https://doi.org/10.1038/s41586-023-06185-3.

Degrave J. *et al.*, 2022, *Magnetic control of tokamak plasmas through deep reinforcement learning.*, «Nature», 602, 7897, pp. 414, https://doi.org/10.1038/s41586-021-04301-9.

Jumper J. *et al.*, 2021, *Highly accurate protein structure prediction with AlphaFold.,* «Nature», 596, 7873, pp. 583, https://doi.org/10.1038/s41586-021-03819-2.

Samuel A.L., 1959, *Some Studies in Machine Learning Using the Game of Checkers*, «IBM Journal of Research and Development», 3, 3, pp. 210, https://ieeexplore.ieee.org/document/5392560.

Shannon C.E., 1950, *XXII. Programming a computer for playing chess*, «The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science», 41, 314, pp. 256, https://vision.unipv.it/IA1/ProgrammingaComputerforPlayingChess.pdf.

Silver D. *et al.*, 2017, *Mastering chess and shogi by self-play with a general reinforcement learning algorithm*, arXiv preprint, https://doi.org/10.48550/arXiv.1712.01815.

Silver D. *et al.*, 2018, *A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.,* «Science», 362, 6419, pp. 1140, DOI: 10.1126/science.aar6404.

Tesauro G., 1995, *Temporal difference learning and TD-Gammon*, «Communications of the ACM», 38, 3, pp. 58, https://dl.acm.org/doi/10.1145/203330.203343.

Tunyasuvunakool K. *et al.*, 2021, *Highly accurate protein structure prediction for the human proteome.,* «Nature», 596, 7873, pp. 590, https://doi.org/10.1038/s41586-021-03828-1.

Turing A.M., 1950, *Computing Machinery and Intelligence*, «Mind», 49, pp. 433, https://academic.oup.com/mind/article/LIX/236/433/986238.

Sitografia

[a] Storia dell'Intelligenza Artificiale, dal corso di "History of Computing" dell'Università di Washington: https://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf.
[b] Per approfondire lo sviluppo dei calcolatori che hanno preceduto, e poi accompagnato, l'IA: https://dataconomy.com/2022/05/16/precursors-of-artificial-intelligence/.
[c] Iris dataset: https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data.
[d] Canale Youtube di Code Bullet: https://www.youtube.com/@CodeBullet/videos.
[e] Pagina web di DeepMind dedicata agli ultimi progressi di AlphaFold: https://deepmind.google/discover/blog/a-glimpse-of-the-next-generation-of-alphafold/
[f] Pagina web del progetto ITER: https://www.iter.org/