

Filosofia e informatica. Il ruolo della logica nell'IA

di Vincenzo Fatigati

Assistiamo a un apparente paradosso: più parole come “intelligenza artificiale”, “automazione” e “algoritmo” entrano a far parte del nostro vocabolario quotidiano, e più il loro significato (e funzionamento) sembra appartenere a una materia esotica. Un algoritmo determina quale notizia compare sulla homepage del nostro social network. Un algoritmo ci suggerisce il nostro possibile libro o il ristorante preferito, in base alle nostre precedenti ricerche. Certi tipi di algoritmi determinano quale e-mail va filtrata nello spam della nostra posta elettronica. Un algoritmo protegge le nostre transazioni bancarie, oppure, ritornando a temi d'attualità, addirittura può determinare la vittoria della campagna elettorale del prossimo presidente degli Stati Uniti d'America¹ o la criticità di una regione in periodo di pandemia. Alla base di tutto ciò si situa la rivoluzione del *machine learning* (“apprendimento automatico”). Per apprendimento automatico si intende la capacità da parte dei computer di apprendere ed eseguire compiti specifici senza essere programmati per farlo, riconoscendo degli schemi tra i dati.

Ancora prima di definire cosa sono questi algoritmi di *machine learning*, qual è la logica interna e la loro possibile classificazione, in maniera analoga si potrebbe sostenere che anche un certo dibattito filosofico contemporaneo è segnato dalla medesima dimensione paradossale: più queste tematiche sembrano assumere un carattere epocale e più lo sguardo del dibattito si allontana dal profilo contenutistico verso quello letterale. Nel primo volume della prestigiosa rivista *Journal of Symbolic Logic*, pubblicato nel 1936, gli articoli si dividevano simmetricamente tra matematici e filosofi: gli studiosi dei due schieramenti, esperti di logica, percepivano di essere accomunati da una medesima impresa conoscitiva. A partire dagli anni Sessanta progressivamente gli articoli scritti da matematici acquistano più spazio, fino ad arrivare agli anni Settanta, quando difficilmente gli articoli scritti dai filosofi venivano accettati dalla rivista. Le due discipline sembravano a tal punto incomunicabili che, per rispondere a quei problemi filosofici ignorati dalla rivista, fu fondato nel 1972 il *Journal of Philosophical Logic*.

¹ Alle elezioni presidenziali del 2012 negli USA, mentre il repubblicano Mitt Romney puntava, col suo sondaggista Neil Newhouse, su un approccio tradizionale, cercando di conquistare gli elettori indipendenti dell'Ohio, il democratico Barack Obama scelse come capo della sua campagna elettorale lo scienziato Rayid Ghani, esperto di *machine learning*, che operò la più grande analisi di dati della storia, realizzando ogni notte 66000 simulazioni di voto tra tutti i dati raccolti. Nonostante Romney prese il 7% in più tra i voti degli indipendenti, perse in quasi tutti gli stati indecisi. Per la vittoria di Obama l'utilizzo del *machine learning* risultò decisivo.

Quest'articolo vuole mostrare nel merito qualcuno di questi "problemi filosofici" rilevanti nel dibattito sull'intelligenza artificiale. Tuttavia, per dirla con le parole di Luciano Floridi (2020), tali "problemi filosofici" non sono sempre sovrapponibili ai problemi dei filosofi: la fecondità dei secondi andrebbe letta alla luce dei primi. Gran parte della letteratura filosofica tradizionale ha posto il problema in maniera dicotomica: ritendendo la tecnica (τέχνη) come blocco monolitico (Martin Heidegger, Hans Jonas, Arnold Gehlen, Günther Anders), e ponendo quindi un discorso filosofico a partire dalla relazione della tecnica con la natura umana o con la scienza. La narrazione postumanista, per esempio, teorizza l'ibridazione delle due nature, umana e artificiale. Altre narrazioni filosofiche, si muovono all'interno di una cornice heideggeriana, ribaltando la prospettiva classica secondo cui la tecnica è un mezzo (prodotto dell'Uomo), e sostenendo invece l'opposto: l'umano come prodotto della tecnica.

Con il passaggio da una tecnologia analogica a una digitale, sempre di più si fa riferimento, nel profluvio eterogeneo di riflessioni su questi temi, all'avvento di un cambiamento paradigmatico. E quindi al passaggio a una nuova era: quella dell'informazione. Si comprende come la riflessione intorno alla scomparsa del lavoro e alla progressiva automazione, anche del settore primario e secondario, è direttamente collegato al modo in cui le tecnologie riducono la forza lavoro necessaria alla produzione. Ovvero, alla capacità di simulare l'intelligenza umana. Questo cambiamento paradigmatico non va solo letto solo in maniera univoca a partire dalle conseguenze dell'innovazione tecnologica declinata secondo un ambito specifico di riferimento politico (es. Big data, social economy); ma a una più attenta lettura questo cambiamento va altresì a ribaltare e ibridare una distinzione classica tra arti liberali e arti meccaniche (o, se si vuole, tra *πρᾶξις* e *ποίησις*) e quindi, come vedremo, tra parola (*λόγος*) e oggetto materiale. Nel suo ultimo libro Bodei (2019) offre una mappatura di questa rivoluzione paradigmatica partendo dal nuovo modo in cui viene declinata la razionalità computazionale².

Aristotele ha teorizzato e giustificato la schiavitù per natura che, a differenza di quella economica, era inscritta nella stessa natura raziocinante dello schiavo, considerato un uomo incompleto e imperfetto, a metà tra oggetto e persona³. Allo schiavo non manca l'intelletto, inteso come la capacità di comprendere ordini. Manca

² Il riferimento che ha sollecitato alcune di queste riflessioni sulla razionalità, prima ancora della pubblicazione del libro *Dominio e sottomissione*, è stato il seminario tenuto dello stesso Bodei "Dallo schiavo al robot. Lavoro, macchine, automazione", all'Istituto Italiano per gli Studi Filosofici (Napoli, 18-20 giugno 2018), a cui va il mio ringraziamento per la qualità del dibattito.

³ Si veda a tal proposito l'incipit della *Politica* di Aristotele [1:1,2]: «[...]L'essere che può prevedere con l'intelligenza è capo per natura [δυνάμενον τῇ διανοίᾳ προορᾶν ἄρχον φύσει], è padrone per natura, mentre quello che può col corpo faticare, è soggetto e quindi per natura schiavo: perciò padrone e schiavo hanno gli stessi interessi». Ovvero, nell'ottica aristotelica vi è simmetria tra chi comanda e chi è comandato: il padrone e lo schiavo sono due figure reciprocamente indispensabili per l'autoconservazione. Lo schiavo, però, rispetto al padrone, manca della capacità deliberativa, cfr. *Politica* [1:13,17]: «[...] perché lo schiavo non possiede in tutta pienezza la parte deliberativa [τὸ βουλευτικόν]. Difatti, come Aristotele puntualizza nel libro VII, al paragrafo 7, ci sono popoli con

piuttosto la capacità deliberativa di autogoverno e di indirizzo (τὸ βουλευτικόν). Da questa valutazione deriva che lo spazio di appartenenza dello schiavo era quello domestico (οἰκία) in contrapposizione allo spazio pubblico politico (πόλις), dove si esercitava proprio quella capacità deliberativa che apparteneva ai cittadini, in accordo con la visione teleologica dello Stagirita. Lo schiavo non poteva quindi essere pienamente felice e realizzato, perché mancava dell'aspetto contemplativo. Se utilizziamo queste categorie aristoteliche, allora possiamo asserire che quella capacità di preveggenza che mancava agli schiavi (anche nei termini di saggezza pratica, φρόνησις) è posseduta invece dall'intelligenza artificiale, la quale non si limita a semplice esecutrice di ordini, ma è dotata di algoritmi che in base al contesto esterno possono apprendere e indirizzare scelte migliori.

Difatti, osservando il funzionamento di una stampante 3D possiamo affermare che il logos diventa corpo materiale, si fa oggetto. Se la distinzione tra padrone e schiavo, nell'ottica aristotelica, è una distinzione tra chi detiene logos, cioè capacità auto-organizzativa, e chi invece ha solo la capacità di obbedire ed eseguire ordini, allora possiamo sostenere che le macchine attuali tramite il *machine learning* possiedono quella stessa facoltà richiesta ai padroni. Pertanto, con la rivoluzione digitale non solo si è radicalizzato ciò che era possibile intravedere già nella prima rivoluzione industriale (cioè, che l'introduzione delle macchine ha aumentato la produzione riducendo la necessità della forza lavoro umana), ma oggi siamo di fronte alla possibilità della sostituzione della forza artificiale con quella umana in diversi settori, grazie a programmi che riescono ad autoprogrammarsi, dando prova di capacità previsiva.

In questo ginepraio di considerazioni filosofiche, bisogna per un attimo dimenticare le grandi costruzioni e trovare il bandolo della matassa in un problema posto da Alan Turing nel 1946, quando – prima ancora che fosse realizzato il primo modello moderno di computer – venivano poste le basi teoriche che hanno poi permesso la realizzazione della macchina computazionale. Turing si chiedeva in che modo fosse possibile parlare di “macchine pensanti”; e oggi, alla luce dell'automazione e della capacità previsiva delle macchine, possiamo chiederci lo stesso.

Le macchine possono pensare?

Proviamo a inquadrare questa domanda, proponendo una sua diversa variante: “*possono gli aeroplani davvero volare?*”. Per rispondere positivamente a questa domanda, cerchiamo prima sul dizionario la definizione di “volare”. Dal *Vocabolario Treccani* si legge la seguente definizione: «Sostentarsi e spostarsi nell'aria con le ali, o con organi di volo analoghi». Se quindi riteniamo che gli

una naturale intelligenza ma che, a differenza dei Greci mancano di quel coraggio necessario che gli permette di avere un'organizzazione sociale libera.

aerei abbiano strumenti di volo analoghi alle ali, affermiamo che effettivamente volano. Chiaramente la risposta dipende dalla nostra inclusione di “organi di volo analoghi” nella definizione, piuttosto che dalla struttura degli aeroplani. Ponendo la domanda in questi termini, può nascondersi un’insidia: confondere il mezzo col fine. Se il nostro fine è costruire agenti che volano, allora ci sono due metodi per realizzare ciò: uno è studiare diversi agenti volanti (uccelli, insetti, pipistrelli ecc.), e quindi concludere che ciò che accomuna tutti questi agenti sono le ali, cercando di riprodurre un modello analogo per l’uomo, come fece Icaro utilizzando un paio di ali; l’altro è studiare l’essenza del volo, quindi l’aerodinamica, e costruire modelli – anche diversi da quelli naturali – ma che ci permettono di volare. Il fine non è mimare gli uccelli, quanto piuttosto volare, cioè simulare quella competenza. Analogamente, diversamente da come ritengono taluni, il fine dell’intelligenza artificiale non è quello di creare umanoidi (cioè, di copiare l’intelligenza umana), quanto di simulare certi comportamenti intelligenti. Quanto poi questo comportamento simulato sia simile alla mente è una questione che passa in secondo piano.

Se quindi il nostro discorso filosofico non vuole seguire l’infausto percorso tracciato da Icaro, facciamo un passo indietro e invece di porre una domanda in generale sull’intelligenza, andiamo in particolare a comprendere il meccanismo delle prime macchine calcolabili. Il merito filosofico di quello che alla storia è passato sotto il nome di test di Turing, oltre all’aspetto tecnico, è stato anche di aver posto il problema sui giusti binari: cioè di aver indirizzato la ricerca più sulla possibilità di simulare comportamenti intelligenti, che sulla definizione o classificazione dei diversi tipi di intelligenza. Diversamente da quanto si pensa, il test di Turing originariamente non voleva fissare le condizioni necessarie o sufficienti della definizione di intelligenza. Un utilizzo del genere è stato fatto dal famoso esperimento di Searle della stanza cinese (1980): è un controesempio mentale per confutare l’ipotesi dell’intelligenza artificiale forte, secondo cui un programma non simula semplicemente l’intelligenza umana (IA debole), ma può essere proprio paragonata a una mente. Searle utilizza il test di Turing come un criterio sufficiente per definire un’intelligenza artificiale: ipotizza che esista una macchina che riesca a generare risposte corrette a ogni domanda che gli viene posta in cinese, in modo che un osservatore esterno non riesca a riconoscere se si tratti di una macchina o di un’intelligenza umana. Secondo Searle, un programma del genere, anche se riuscisse nell’intento della trascrizione, non saprebbe parlare cinese, semplicemente perché funziona sul livello sintattico, che non è equivalente a quello semantico (lui stesso, posto all’interno del meccanismo, potrebbe eseguire quelle operazioni di trascrizione senza capire nulla di cinese). Senza entrare in merito nel dibattito decennale, è opportuno sottolineare a tal riguardo che Turing non ha mai usato il termine test ma quello di *imitation game*; e in ogni caso Turing non lo intendeva come criterio sufficiente e neanche necessario per l’attribuzione di intelligenza. Infatti Turing si è limitato a parlare di gioco di imitazione, che non si riferisce a singole capacità. E

inoltre nel gioco di Turing la prova da superare non è da parte delle macchine ma degli interroganti (che non sono esperti); in ogni caso, tale superamento non implica che la macchina abbia quella capacità.

Abbandonando il dibattito sull'intelligenza artificiale forte, cerchiamo invece di comprendere meglio i limiti computazionali. Entriamo quindi nel cuore del problema, e a questo punto chiediamoci se, con le nostre macchine super potenti, sia possibile costruire un computer che risolva come una calcolatrice tutti i problemi matematici (e non logici). Detta in maniera un po' più formale: se consideriamo una teoria del primo ordine, come quella dell'aritmetica (T_A) la cui struttura è $A = (\mathbb{Z} \mid +, \cdot, 0, 1) -$ o una teoria ugualmente potente come la teoria sui numeri T_n la cui struttura è $N = (\mathbb{N} \mid +, \cdot, 1) -$, vogliamo scoprire se esiste un algoritmo (ALG) tale che per ogni proposizione in T_n (o T_A) ci mostra un "sì" di output se e solo se $T_n \vdash \phi$. Vedremo che tale algoritmo non esiste, il che equivale a dire che esisteranno proposizioni nel sistema che sono indecidibili (non dimostrabili o refutabili). Per risolvere il quesito, prendiamo il dizionario che abbiamo lasciato all'inizio del paragrafo, e consideriamo la seguente proposizione:

[a] "Esiste un minimo numero naturale che non può essere definito utilizzando al massimo 20 lettere del *Dizionario* Treccani".

Ora, proviamo a chiederci se la proposizione [a] sia vera o falsa. La negazione di [a] non può essere vera, perché implicherebbe che tutti i numeri possono essere definiti al massimo utilizzando 20 parole. Ma ciò è impossibile, perché nel nostro dizionario ci sono solo un numero finito di parole che hanno al massimo 20 parole. D'altro canto [a] non può essere vera, perché se fosse così si auto contraddice: dato che [a] rappresenta tale numero con meno di 20 parole. Abbiamo quindi un'antinomia, non possiamo né affermare né refutare [a]. Cerchiamo di "tradurla" in termini matematici. Allora invece di riferirci alle parole troviamo un sistema di codifica: contiamo i numeri dei simboli che ci sono nella nostra formula (V_n) includendo parentesi, variabili. Quindi la lunghezza della nostra (V_n) formula è rappresentata dal numero dei simboli che ci sono. Pertanto quando noi diciamo che il numero n è definito da una V_n formula $\phi(x)$ con la lunghezza limite di l , noi intendiamo codificarla nella formula $A' (n, l)$ con n e l variabili libere.

$$\mathbf{N} \models \phi(a) \text{ se e solo se } a = n$$

Kurt Gödel, col suo famoso teorema di incompletezza, dimostra che $A' (n, l)$ non può essere espressa nella logica del primo ordine, cioè non esiste una V_n formula $\psi(x, y)$ tale che $\mathbf{N} \models \psi(n, l)$ se e solo se è definibile da una formula con la lunghezza massima di l . Questo significa che non esiste un algoritmo che possa determinare, per una formula arbitraria, se è vera o falsa nella nostra teoria T_n . Cioè necessariamente esiste una proposizione T_n per cui il nostro algoritmo non computa la risposta corretta. Detta diversamente, il primo teorema di

incompletezza afferma che non esiste un sistema consistente di assiomi i cui teoremi possono essere enumerati da un algoritmo capace di provare tutte le verità aritmetiche dei numeri naturali. Per ogni sistema formale del genere ci sarà sempre una proposizione che sarà vera nel sistema ma non dimostrabile all'interno. Il secondo teorema di incompletezza, invece, mostra che tale sistema non può dimostrare la sua consistenza.

Senza entrare in altri tecnicismi, l'importanza di questa strategia dimostrativa consiste nell'aver aritmetizzato (codificato tramite l'aritmetica) concetti metalinguistici, come quello di "sintassi", autoreferenzialità, o lo stesso concetto di "dimostrazione", in modo da costruire a sua volta una dimostrazione che pur seguendo la struttura antinomica del famoso paradosso del mentitore (*questa frase è falsa*) aggira la conclusione paradossale, perché nella versione di Gödel si fa riferimento alla dimostrabilità stessa (*questa frase non è dimostrabile*). Cioè, si arriva a dimostrare che esistono proposizioni vere ma non dimostrabili, evitando la circolarità del paradosso. Fu proprio studiando il teorema di Gödel che Turing elaborò la prima idea di macchina di Turing, arrivando successivamente a equivalenti conclusioni di indecidibilità nel campo informatico degli stessi teoremi limitativi, dimostrando operazioni che la macchina non può intrinsecamente eseguire. Ad esempio, scoprì sia il teorema della terminazione (considerato un programma e un input non è possibile stabilire se il programma termina oppure continuerà l'esecuzione all'infinito), e anche lo stesso problema della decisione proposto da David Hilbert (*l'Entscheidungsproblem*, ovvero scoprire che per ogni formula della logica nel primo ordine non esiste un algoritmo in grado di stabilire se essa fosse valida o meno).

L'idea della macchina di Turing era piuttosto semplice: consiste in un prototipo mentale di una macchina che, in base a determinate istruzioni (programma), attraverso un nastro potenzialmente infinito riesca a manipolare questi simboli secondo le istruzioni contenute nel programma. Turing successivamente comprese che fosse possibile elaborare una macchina di Turing che poteva simulare una qualsiasi macchina di Turing dato un arbitrario input; un po' come accade ai nostri computer che in base a determinati programmi riescono a decodificare e simulare le istruzioni del programma. A queste macchine diede il nome di macchine universali. Comprese quindi che, oltre alla struttura delle macchine (quello che oggi chiameremmo *hardware*), bisognava considerare il programma, l'insieme delle istruzioni che andiamo a dare alla macchina (il *software*). Non era sufficiente potenziare l'hardware per eseguire diversi tipi di calcolo. La macchina universale di Turing era quindi una macchina capace di emulare e simulare qualsiasi programma, proprio come il computer (che, tecnicamente è una macchina potenziale di Turing, giacché la memoria non è infinita).

Quindi, coi risultati limitativi di Gödel e Turing siamo arrivati a uno sguardo sui limiti intrinseci di un sistema formale. Risultati che, come affermò lo stesso Gödel nelle conversazioni con Hao Wang, trovano un'analogia nei limiti della ragione di kantiana memoria: se la ragione vuole essere coerente, cioè non

contraddittoria (non cadere in antinomia), allora deve essere limitata; se invece vuole essere completa e trattare le idee trascendentali, allora cade in contraddizione. La completezza implica inconsistenza: proprio quello che accade se accettiamo l'esistenza del nostro algoritmo nel sistema formale. Naturalmente partendo dalla constatazione di questi limiti formali si possono fare diverse valutazioni, sia sul profilo più psicologico che su quello più filosofico. Ad esempio, già Cellucci (1998) concludeva, anche alla luce dei teoremi limitativi degli anni Trenta, che la logica matematica avrebbe dovuto abbandonare l'impianto ideologico positivistico su cui è stata codificata, nel periodo che va dall'*Ideografia* di Frege (1879) ai *Fondamenti di Logica Teoretica* di Hilbert (1928), per adottare invece sistemi modulari di conoscenza, cioè gestire l'informazione in sistemi "aperti", mantenendo l'interazione e la connessione con altri saperi. Cellucci concludeva che tali limiti dei sistemi formali dimostrerebbero al contrario, l'importanza di un atteggiamento euristico, e non dogmaticamente legato a sistemi chiusi, come quelli dei sistemi formali deduttivi. In questo senso lo sviluppo recente dell'informatica dimostrerebbe l'importanza del metodo analitico rispetto a quello assiomatico.

Oltre che per le ragioni su menzionate, i teoremi limitativi ci hanno portato a individuare una differenza tra il livello teorico e quello tecnologico. Senza affrontare il tema della teoria della complessità, si potrebbe dire che dal 1946 a oggi non vi è stato nessun progresso a livello teorico sulle macchine, ma solo di tipo tecnologico. Il modello di Turing, anzi, ha anticipato sul piano teorico cioè che è successivamente è stato realizzato a livello tecnologico. Il nostro smartphone, il computer dell'ultima generazione o probabilmente il computer quantistico del futuro possono potenzialmente eseguire tutte e sole le operazioni eseguibili da macchine di Turing universali. Ovviamente la differenza è che oggi i nuovi dispositivi di calcolo sono più accessibili e molto più piccoli e veloci. Ma sono tutte potenziali macchine universali di Turing. Al punto che oggi la definizione accettata di "calcolabile" viene inteso proprio come "risolvibile da una macchina di Turing universale".

Se non vogliamo cadere nella circolarità di tale definizione, dobbiamo quindi chiederci cosa si intende esattamente per calcolare funzioni calcolabili. Prima di morire lo stesso Turing si chiese fino a che punto le macchine potessero simulare l'intelligenza umana. Per rispondere all'argomento elencò alcune facoltà che non potevano essere simulate, tra cui essere gentile, essere bello, avere il senso dell'umorismo, imparare dall'esperienza, far innamorare qualcuno e così via. Tra queste la previsione sbagliata di Turing riguarda principalmente la capacità di apprendere dall'esperienza. Il giocatore di scacchi David Levy ha effettivamente ipotizzato che per il 2050 ci innamoreremo di androidi. Ipotesi che, benché non ancora plausibile in campo scientifico, ha trovato già realizzazione nella cinematografia. Il modo migliore per spiegare questo cambiamento paradigmatico, e quindi per rispondere alla domanda di Turing, è forse proprio partire dai giochi da tavolo.

Come pensano le macchine

Alpha Go zero è un software che è riuscito a sconfiggere campioni di un gioco piuttosto complesso, il Go, senza che fosse stata specificamente programmato per farlo: ha appreso le regole del gioco giocando numerose partite contro sé stesso, e quindi apprendendo come gli umani dall'esperienza. Naturalmente non è nuova l'idea di programmare un computer in grado di elaborare strategie vincenti di un gioco: già a partire dagli anni Cinquanta si è provato a elaborare un programma che simulasse il gioco del tic-tac. Oppure, già nel 2010 sono stati elaborati super computer con capacità fino a qualche decennio fa impensabili: Watson, computer elaborato dall'IBM, comprende il linguaggio naturale riuscendo a rispondere correttamente alle domande di un famoso quiz televisivo battendo gli altri concorrenti. Ancora, ricordiamo come negli anni Novanta fece scalpore l'impresa di Deep Blue, che sconfisse il campione di scacchi Garry Kasparov. Eppure, con Alpha Go e Alpha Go zero, i due computer programmati da Google, si è fatto un salto paradigmatico nell'intelligenza artificiale. La vera differenza consiste non nello scopo della loro programmazione, ma nel modo in cui sono stati programmati.

Nella prima versione, Alpha Go ha imparato ad avere un'intelligenza sovrumana giocando un certo numero di partite contro esperti. Nell'ultima versione del 2017, Alpha Go zero è invece arrivato a tali livelli senza nessuna forma di programmazione esterna in merito al gioco e senza nessuna simulazione di partite contro campioni, ma apprendendo semplicemente giocando milioni di volte contro sé stesso. Inoltre, mentre il gioco degli scacchi è un gioco finito, nel senso che è possibile registrare su un database tutte le possibili mosse programmando di volta in volta la scelta migliore per ogni mossa, il gioco cinese del Go non è un gioco finito: e quindi ha una complessità di variabili maggiore. Deep Blue, il computer che sconfisse il campione del mondo Kasparov, non si servì di un database, ma era programmato da un albero di ricerca, in grado di elaborare la mossa più vantaggiosa durante la partita. Era programmato a eseguire questa funzione nel modo migliore. Invece con Alpha Go assistiamo a una struttura di programmazione del tutto diversa: la strategia migliore si fonda sul *machine learning* e sulle reti neurali, apprendendo in base all'esperienza strategie sempre migliori in modo da superare qualsiasi intelligenza umana, senza avere nessuna programmazione specifica a quel compito! Proprio come avviene con l'apprendimento degli umani.

Nella prima parte di questo articolo è stato introdotto l'argomento, spiegando come il cambiamento paradigmatico della nuova rivoluzione consista proprio nella capacità da parte dell'intelligenza artificiale di fare previsioni su futuro, di riuscire a trovare e modellare schemi raffinati a partire da una quantità di dati. Nella seconda parte è stata ribaltata la prospettiva, affrontando l'argomento da un livello più teorico per comprendere effettivamente le potenzialità e limiti delle macchine calcolabili, riducendo eventuali interpretazioni fuorvianti. Ora possiamo affrontare il problema di come e in che senso le mac-

chine possono prevedere, e quindi imparare in base all'esperienza partendo da una branca specifica dell'intelligenza artificiale: il *machine learning*.

La rivoluzione del *machine learning* consiste nell'inversione del processo di deduzione: si parte dai dati del risultato desiderato per arrivare agli algoritmi. Si elaborano algoritmi di apprendimento (*learner*) capaci di elaborare altri algoritmi. In un certo senso i computer scrivono altri programmi. L'algoritmo, cioè (come abbiamo visto) l'insieme di regole che andiamo a dare a una macchina, può a sua volta modificarsi e creare autonomamente altri algoritmi, altre istruzioni per la macchina. Affrontare, anche solo minimamente e a livello panoramico, questa corposa branca in questa sede è ovviamente impresa chimerica. Esistono diverse scuole, spesso anche contrastanti all'interno del mondo del *machine learning*. Ci sono i simbolisti, secondo cui l'apprendimento è un processo inverso della deduzione, e si ispirano alla logica alla filosofia e alla psicologia; i connessionisti, invece, si ispirano alle neuroscienze e alla fisica per elaborare i propri algoritmi di apprendimento; gli evoluzionisti alla genetica e alla biologia evolutiva per le simulazioni numeriche dell'evoluzione; i bayesiani, formano la loro base di apprendimento sulla statistica, e quindi un'inferenza probabilistica; e infine gli analogisti fondano la loro struttura di conoscenza sulle analogie (attraverso psicologia e ottimizzazione matematica).

I paradigmi di apprendimento sono tre, corrispondenti a un particolare funzione: supervisionato, non supervisionato e per rinforzo. L'apprendimento *supervisionato* è un tipo di apprendimento che si basa sulla previsione del valore di output partendo da un numero ristretto di esempi corrispondenti, tipo coppie di valori input-output. Quindi l'apprendimento supervisionato, in base ai nuovi dati e a un'efficace generalizzazione, permette di migliorare la relazione iniziale tra dati di ingresso e uscite (es. dato T come insieme di addestramento, la nostra funzione elabora man mano i dati cercando di ottimizzare le previsioni del rapporto tra input e output).

A differenza dell'apprendimento supervisionato, l'apprendimento *non supervisionato* non associa alcuna particolare funzione di apprendimento all'input. Semplicemente, partendo da certi dati $\{x_1, x_2, \dots, x_n\}$, cerca di estrapolare dati interessanti, analizzando certe regolarità (*data mining*). Quindi, attraverso metodi topologici e probabilistici, gli algoritmi d'addestramento non supervisionati fanno solo riferimento alle variabili d'ingresso, cercando di estrapolare cluster rappresentativi (un po' come con i motori di ricerca che reindirizzano a diversi link). Infine, nell'apprendimento *per rinforzo* la macchina riceve informazioni dall'esterno, da un determinato ambiente, e in base a feedback positivi o negativi migliora il proprio algoritmo: individuando quindi un *modus operandi* partendo dalla somma dei feedback positivi, come abbiamo visto nel caso di giochi che migliorano le loro prestazioni in base al numero di partite che hanno giocato. L'apprendimento per rinforzo non elabora quindi coppie di input-output di esempi, ma in base alle informazioni esterne elabora in maniera ottimale il proprio algoritmo.

Che tipo di ragionamento noi esseri umani eseguiamo quotidianamente? Possiamo formulare la nostra domanda meglio: se vogliamo fare previsioni in maniera più accurata, per esempio nella diagnosi di una malattia, nel tentativo di ricordarci dove abbiamo lasciato le chiavi della macchina la scorsa notte, o nella scoperta di un caso irrisolto di omicidio, che tipo di ragionamento effettuiamo? Secondo Arthur Conan Doyle, il padre putativo di Sherlock Holmes, per la soluzione brillante di gialli irrisolti da parte dell'investigatore la risposta corretta è "utilizza un metodo deduttivo, preciso come le proposizioni di Euclide". In realtà, il ragionamento di Sherlock Holmes non è deduttivo, e neanche induttivo, ma una sorta di abduzione. Prima di spiegare questa famiglia di logiche (non monotoniche) che riescono a formalizzare questo tipo di ragionamenti, è utile partire dal famoso esempio dei fagioli di Pierce per comprendere la differenza tra questi tipi di pensiero. In realtà il rapporto sul il ragionamento abduttivo di Pierce e Sherlock Holmes è dibattuto. Secondo Cellucci (Springer, forthcoming) il metodo abduttivo, a differenza del metodo analitico, non amplia la conoscenza.

Deduzione

Regola: Tutti i fagioli di questo sacchetto sono bianchi.

Caso: Questi fagioli vengono da questo sacchetto.

Risultato: Questi fagioli sono bianchi.

Induzione

Regola: Questi fagioli vengono da questo sacchetto.

Caso: Questi fagioli sono bianchi.

Risultato: Tutti i fagioli di questo sacchetto sono bianchi.

Abduzione

Regola: Tutti i fagioli di questo sacchetto sono bianchi.

Caso: Questi fagioli sono bianchi.

Risultato: Questi fagioli vengono da questo sacchetto.

Quindi, quando a lezione di geometria euclidea abbiamo imparato a dedurre attraverso regole logiche delle conclusioni (teoremi) a partire da determinate premesse (assiomi), abbiamo utilizzato un metodo deduttivo: la conclusione segue necessariamente dalle premesse. Il secondo tipo di ragionamento è quello induttivo (nel senso di Hume), che è il ragionamento proprio del metodo scientifico: partendo dall'osservazione di casi particolari si cerca di inferire una legge generale. Ovviamente l'induzione ha un valore di certezza più debole, e quindi probabilistico rispetto alla precedente. Il terzo tipo di ragionamento è

abduittivo, simile al tirare a indovinare: come capita quando, in un romanzo poliziesco, pensiamo (spesso erroneamente) che il maggiordomo trovato nella stanza del delitto sia il colpevole (cioè, in base all'esperienza inferiamo l'ipotesi più probabile che giustifica quell'esperienza).

Queste brevi riflessioni possono applicarsi anche ai linguaggi di programmazione e alle regole che inseriamo nei nostri computer. Cerchiamo di comprendere con qualche esempio (che non sia il funerario esempio degli uomini mortali e di Socrate) la logica sottostante. Consideriamo un database in cui sono inseriti i nomi e le relazioni della famiglia di Platone. Nelle nostre premesse inseriamo i dati genealogici della madre e dei fratelli del noto filosofo. Il linguaggio di programmazione utilizzato è Prolog, in quanto è quello più simile alla logica matematica. Nell'esempio riportato, infatti, 'madre', 'genitore', 'nonni' sono relazioni con numero di varietà 2. Nominando le relazioni solo con l'iniziale in maiuscolo, avremo che $\forall xM(x,y) \rightarrow G(x,y) \text{ e } \forall x(M(x,z) \wedge G(z,y)) \rightarrow N(x,y)$.

```
madre(perictione,platone).
madre(perictione,glaucone).
madre(perictione,adimanto).
genitore(X,Y):- madre(X,Y).
nonna(X,Y):- madre(X,Z),genitore(Z,Y).
```

In questo caso i primi tre sono fatti che accettiamo essere veri(assiom), le altre due sono le regole che andiamo a utilizzare per il nostro database. Il database registra che Perictione è la madre di Platone. Le ultime due sono regole che andiamo a inserire: 1) se qualcuna è madre di qualcun'altra, allora è anche genitore; 2) se X è madre di Z, e a sua volta Z è genitore di Y allora X sarà la nonna di Y. In questo modo possiamo interrogare il nostro programma con una semplice domanda: Perictione è genitore di Platone? E il nostro computer, con un processo di logica deduttiva, inferisce la soluzione:

```
?- genitore(perictione,platone)
true.
```

Possiamo aggiungere nuove regole (ad esempio, "se due persone condividono gli stessi genitori, allora sono fratelli/sorelle"), e a nostra volta aggiungere altre informazioni nel database, per esempio su nipoti o figli di figli: il programma funziona applicando semplicemente le stesse regole logiche conosciamo. Per esempio, se dovessimo chiedere "chi sono i figli di Perictione" basta digitare

```
? genitore(perictione, X)
X= platone
X= glaucone
```

E in base alle informazioni che inseriamo nel database il compilatore ci dirà il nome dei figli. Questo processo è deduttivo proprio come nei teoremi di Euclide.

Logica induttiva

Nella logica induttiva, invece, il nostro percorso è inverso: cerchiamo l'algoritmo partendo dai dati. Cioè dobbiamo trovare le teorie che giustificano i dati che possediamo. Abbiamo quindi una conoscenza di sfondo B (*background knowledge*), un insieme di ipotesi positive E^+ e altre invece negative E^- . Il nostro ragionamento deve essere proprio quello di trovare l'ipotesi H (programma logico) tale che spieghi tutti i fatti positivi e nessuno dei negativi: cioè, H tale che H e B implicano E^+ e H e B non implicano E^- . Cioè per ogni $e \in E^+ : B \cup H \models e$ e quindi $\forall e^- \in E^- : B \cup H \not\models e^-$. Proviamo a formalizzarlo nel nostro programma, considerando la determinata relazione di parentela: Anna e Ugo sono i nonni rispettivamente di Maria e Gigi. Indichiamo le costanti individuali con la lettera iniziale del nome di persona corrispondente:

nonni(a,m).
 nonni(a,g).
 nonni(u,m).
 nonni(u,g).

A questo possiamo aggiungere ulteriori informazioni, come in B (*background knowledge*): andiamo cioè a indicare che Pina è la madre di Maria e di Ugo.

madre(p,m)
 madre(p,g)
 madre(p,g)

In questo caso il nostro programma, a differenza del precedente, deve inferire un'ipotesi H che spieghi la somma di questi fatti, cioè B e E^+ , scartando ovviamente quelli negativi. Avremo quindi:

nonni(X,Y): genitore(X,Z), genitore(Z,Y).
 genitore(X,Y): madre(X,Y).
 genitore(X,Y): padre(X,Y).

In questo caso non siamo stati noi ad aver impostato la regola, come nell'esempio precedente, ma il programma analizzando i diversi dati inferisce l'ipotesi più plausibile: se X è genitore di Z (e tutte le madri di un arbitrario X

sono genitori di X), e a sua volta Z è genitore di Y, allora X sarà la nonna di Y. Il programma cerca cioè di inferire la prova che è valida generalmente fino a prova contraria, e va a integrare gli altri dati utilizzando questa regola appresa confrontando i dati. Ovviamente, il problema sorge quando magari vogliamo specificare se la nostra X è un nonno o una nonna, oppure nel caso in cui ci sono fratelli non germani: il nostro algoritmo in base alle informazioni deve adottare strategie che riescano a includere queste eccezioni. In che modo?

Logiche non monotone

Esiste una differenza fondamentale tra il ragionamento logico che utilizziamo quando cerchiamo di dimostrare i teoremi di matematica (deduttiva), e quella che invece utilizziamo nel nostro ragionamento quotidiano. Nelle nostre dimostrazioni matematiche, se andiamo ad aggiungere altre premesse a quelle note questo non invalida le conseguenze precedentemente dimostrate. Questa proprietà si chiama *monotonia*: aumentando l'insieme delle premesse non diminuisce l'insieme delle conseguenze. Cioè se consideriamo essere l'insieme delle nostre premesse tale che $\Gamma \subset \Delta$ e $\Gamma \models \phi$ allora avremo anche che $\Delta \models \phi$. Nel nostro ragionamento quotidiano (ma anche in quello dei medici e detective quando sono a lavoro) formuliamo ipotesi plausibili in base ai nostri dati, e poiché la nostra conoscenza di base è incompleta, le nuove informazioni che abbiamo ottenuto possono farci ritrattare conseguenze che ritenevamo precedentemente plausibili. Quotidianamente utilizziamo quindi una logica dinamica, più che una logica deduttiva. Se dobbiamo formalizzare questi tipi di ragionamento, allora dobbiamo considerare non solo la possibilità di trovare dati che vanno in conflitto con le nostre ipotesi, ma riuscire a elaborare strategie che possano risultare il più possibile adatte a quel determinato contesto, fornendoci la scelta più plausibile, includendo ovviamente, per quanto possibile, tutte le diverse eccezioni.

Forniamo un esempio classico. Se nelle nostre premesse abbiamo (I) *Gli uccelli (generalmente) volano*; (II) *Titti è un uccello*; allora, per la classica regola del *Modus Ponens*, possiamo derivare che (III) *Titti Vola*. Ma attenzione: quel *generalmente* sta a significare “nella maggioranza dei casi”. Quindi a ben vedere la nostra relazione di deducibilità non è classica, cioè la conclusione in (III) non è conseguenza logica. Lo è solo nel caso in cui la conseguenza segua necessariamente dalle premesse. Infatti, se otteniamo una nuova informazione, secondo cui (IV) *Titti è uno struzzo*, poiché sappiamo che gli struzzi non volano, allora invalideremo la conclusione in (III), secondo cui Titti vola. Certo, si può inserire nel nostro database l'eccezione, e quindi formalizzare come sistema deduttivo $[\forall x (Uccello(x) \wedge \neg Struzzo(x)) \rightarrow Vola(x)]$. Ma se vogliamo elaborare un sistema logico dinamico che cerca di catturare il nostro ragionamento quotidiano, allora dobbiamo tenere in conto non solo di quella singola

eccezione, ma di tutte le possibili eccezioni che non sono ancora state inserite nel nostro database. E quindi, quando successivamente la nostra informazione che otteniamo ci informa che Titti è un pinguino, non possiamo dedurre in virtù dell'insieme delle nostre eccezioni che vola, perché sappiamo che anche i pinguini non volano. Pertanto, possiamo considerare un Predicato P che include tutte le possibili eccezioni e anormalità, in modo da invalidare la nostra relazione di deducibilità in tutte queste eccezioni.

Cerchiamo brevemente di riassumere, con questo schema, considerando ovviamente in lettera maiuscola il predicato corrispondente e in minuscola le costanti individuali:

- (1) $\forall x (U(x) \rightarrow V(x))$; Se $U(t)$ allora $V(t)$. [Logica Classica]
- (2) $\forall x (U(x) \wedge \neg S(x) \wedge \dots) \rightarrow V(x)$; Se $U(t)$ allora per derivare $V(t)$ nella logica classica bisogna dimostrare $\neg S(t)$ e tutte le eccezioni che andremo a inserire (...). Procedimento che in base al numero di eccezioni rischia di diventare lungo.
- (3) $\forall x (U(x) \wedge \neg P(x) \rightarrow V(x))$ Se $U(t) \vdash V(t)$. [Ragionamento per default]. Per derivare $V(t)$ non bisogna dimostrare, come nell'inferenza classica, $\neg P(t)$. Lo accettiamo fino a quanto non sarà dimostrato $P(t)$.

In (3), pur sapendo che la nostra relazione di deducibilità non è di tipo classico, la accettiamo fino a prova contraria, cioè finché non sarà comparsa un'informazione che appartiene all'insieme delle eccezioni e ci faccia invalidare la nostra relazione. Un po' come quando nei processi si dice che si è non colpevoli fino a prova contraria. Quest'ultimo aspetto, denominato in inglese "ipotesi del mondo chiuso" (*closed world assumption*, CWA), è molto importante nei linguaggi di programmazione (come Prolog, il linguaggio precedentemente usato). Sicché se l'informazione esterna che otteniamo non è presente nel nostro database, nelle nostre assunzioni di partenza, allora diciamo che è falsa. Cioè, assumiamo che nel nostro Database c'è l'informazione positiva, e tutto ciò che non si deduce è da considerare falso (in questo caso: l'assenza della prova è la prova dell'assenza). Ovvero, se D è il nostro database e CWA la nostra ipotesi di mondo chiuso:

$$D \not\models \phi \text{ allora } D + CWA \vdash \neg \phi$$

L'assunzione dell'ipotesi del mondo chiuso implica una relazione non monotona che può causare qualche problema. Ad esempio, se nel nostro database c'è una disgiuntiva, allora in base a questa ipotesi deriviamo la negazione di ciascun disgiunto, che però in questo modo, essendo entrambe negative (non deducibili), va a contraddire la nostra ipotesi di partenza della verità di un disgiunto. In questo caso si sono elaborati sistemi logici sempre più raffinati che cercano proprio di risolvere questo tipo di contraddizioni, formalizzare il ragionamento quotidiano e le diverse possibilità in base ai dati forniti dall'e-

sterno. Per esempio, la *logica adattiva* di Batens elabora un sistema di logica adattiva secondo cui le regole di inferenza possono valere limitatamente a un certo contesto o un certo tempo. Inoltre fa una differenza tra un sistema dinamico esterno e interno. Nel sistema esterno, quando le informazioni contraddicono le nostre ipotesi di partenza utilizziamo una relazione di deducibilità non monotona. Invece, nei sistemi dinamici interni tale conflitto può avvenire all'interno delle nostre stesse ipotesi, e allora è necessaria una relazione di deducibilità paraconsistente (che invalida principio logico dell'ECQ), cioè tale che possa tollerare livelli di inconsistenza senza diventare incoerente. Quindi si lavora su un insieme di anormalità e su diverse relazioni di deducibilità che valgono in certi tempi e in certi contesti, e che possono tollerare certi livelli di contraddizione, senza far implodere il sistema.

Conclusioni

Gli esempi a cui abbiamo accennato mostrano come la nuova rivoluzione presuppone, oltre che a uno sforzo tecnologico, anche un'impresa teorica. Naturalmente, possiamo concludere continuando a elencare la lista introdotta all'inizio: fabbriche, orari dei voli, elettrodomestici, giochi, automobili, o nuove innovazioni in campo biomedico e bioinformatica sono tutte guidate da questi algoritmi. Al contempo, possiamo elencare le minacce nel campo dell'industria bellica, come i droni automatici, oppure altri strumenti di controllo da parte delle multinazionali che utilizzano le nostre informazioni come petrolio da raffinare per scopi commerciali, di marketing industriale, o anche per finalità politiche. Eppure, per quanto possa sembrare strano, tutti questi calcoli, che nei computer vengono eseguiti da porte logiche realizzate coi transistor, possono essere ricondotti a tre semplici operazioni logiche: AND, OR e NOT.

Sicuramente, compito della filosofia è quello di comprendere il cambiamento paradigmatico in atto, di coglierne le contraddizioni e riuscire a comprendere nuovi i problemi etici e sociali che si stagliano all'orizzonte. Ma non dimentichiamo che, proprio come nella programmazione, il compito filosofico è sempre primariamente quello di comprendere e interpretare il mondo (l'insieme dei fatti) attraverso la manipolazione del linguaggio. E in questo caso il ruolo scientifico rivestito dalla filosofia e dalla logica filosofica può apportare un contributo positivo alla stessa impresa scientifica elaborando nuovi modelli di formalismo su settori non ancora opportunamente esplorati. McCarthy, l'informatico che coniò per la prima volta il termine di "intelligenza artificiale", fu un pioniere, spesso isolato, nel comprendere come lo studio dei programmi informatici fosse intrecciato a problematiche filosofiche di tipo epistemologico: lavorò per tutta la carriera codificare e formalizzare il ragionamento del *senso comune* (cfr. McCarthy, 1995). Le ricerche degli ultimi anni nel campo informatico, spinte sempre più da questa vocazione unitaria dei saperi, hanno dato ra-

gione al padre dell'intelligenza artificiale, chiudendo il cerchio con l'Aristotele che abbiamo lasciato all'inizio dell'articolo. Lo Stagirita, infatti, pur parlando di logica pratica, cioè di un ragionamento che potesse includere anche le eccezioni, non riuscì a formalizzarlo con le inferenze che appartengono alla teoria sillogistica, decretando la separazione della logica da quella vocazione originaria. Gran parte dei ragionamenti che facciamo quotidianamente, infatti, non sono inferenze matematiche; e al contrario, il ragionamento matematico spesso si riduce a mero calcolo. Oggi, però, nel campo informatico è una prateria ancora inesplorata, dove entrambe queste esigenze possono trovare una sintesi: utilizzare il rigore matematico per codificare nuovi domini di ragionamento.

Insomma, parafrasando laicamente l'incipit del Vangelo di Giovanni (*in principio era il Logos*), possiamo affermare, davanti alle complesse ed enormi combinazioni di transistor che eseguono queste gigantesche montagne di algoritmi, che è tutto una questione di logica.

Bibliografia

- Bodei R. *Dominio e sottomissione. Schiavi, animali, macchine, intelligenza artificiale*, il Mulino, Bologna, 2019.
- Bringsjord S., Govindarajulu N.S. *Artificial intelligence*, in Zalta E.N. (a cura di), *The Stanford Encyclopedia of Philosophy*, Metaphysics Research Lab, Stanford University, 2018.
- Cellucci C., *Le ragioni della logica*, Laterza, Bari-Roma, 1998.
- Domingos P., *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books, New York, 2015.
- Eco U., *Il segno dei tre: Holmes, Dupin, Peirce*, Bompiani, 1983.
- Floridi L. *Pensare l'infosfera: La filosofia come design concettuale*, Raffaello Cortina, Milano, 2020.
- Hedman S., *A First Course in Logic: An introduction to model theory, proof theory, computability, and complexity*. Oxford University Press, 2004.
- McCarthy J., *What has AI in Common with Philosophy?* in *IJCAI '95: Proceedings of the 14th International Joint Conference on Artificial Intelligence* vol. 2, agosto 1995.
- Poole D.L., Mackworth A.K., Goebel R., *Computational intelligence: a logical approach* vol. 1, Oxford University Press, 1998.
- Russell S.J., Norvig P., *Artificial intelligence: a modern approach*, Pearson Education Limited, 2016.
- Thomson R., *Logic and Artificial Intelligence*, in Zalta E.N. (a cura di), *The Stanford Encyclopedia of Philosophy*, Metaphysics Research Lab, Stanford University, 2020.