UNIVERSITÀ DEL SALENTO

Corso di Laurea TRIENNALE in INGEGNERIA
DELL'INFORMAZIONE

L'evoluzione dei motori di scacchi: dall'approccio tradizionale all'intelligenza artificiale

Relatore:
Prof.re Italo Epicoco

Correlatore:
Prof.re Massimo Cafaro

Laureando:
Francesco Maggio
Matricola n° 20071278

ANNO ACCADEMICO 2023/2024

"I computer sono incredibilmente veloci, accurati e stupidi. Gli uomini sono incredibilmente lenti, inaccurati e intelligenti. L'insieme dei due costituisce una forza incalcolabile."

- A. Einstein

Indice

1 Gli Scacchi oggi: tra Tradizione e Innovazione **1**

1.1 Introduzione al gioco degli scacchi 1

1.2 Il fascino di un gioco senza tempo 2

1.3 Obiettivi e finalità della tesi 3

2 L'approccio tradizionale dei motori di scacchi **4**

2.1 I limiti della mente umana 4

2.2 L'immensità dell'albero scacchistico 4

2.2.1 Il numero di Shannon 5

2.2.2 Lo "scacchista camionista" 7

2.3 L'avvento dei motori scacchistici 8

2.3.1 I motori di scacchi nascono con l'informatica 8

2.3.2 Cenni sulla teoria dei giochi 9

2.3.3 Turochamp: il primo motore della storia 10

2.4 Il contributo del padre della teoria dell'informazione 12

2.4.1 La formula di Shannon 13

2.4.2 I difetti della valutazione 15

2.4.3 La strategia A 18

2.4.4 La strategia B 20

2.5 La potatura alfa-beta 21

2.5.1 Complessità computazionale 22

2.5.2 Esempio di potatura alfa-beta 23

2.6 L'avvento dei motori scacchistici in Unione Sovietica 25

2.6.1 I russi hanno insegnato scacchi al mondo 25

2.6.2 Il primo motore di scacchi russo 25

2.6.3 La congettura di Botvinnik 27

2.6.4 La guerra fredda . . . sulla scacchiera 28

2.7 La nuova generazione 29

2.7.1 MacHack 29

2.7.2 Kaissa . . . 29
2.7.3 Pioneer . . . 30
2.7.4 Belle . . . 30
2.7.5 Deep Blue . . . 32
2.8 Lo scontro che ha sconvolto la storia . . . 33
2.8.1 Kasparov: il difensore della razza umana . . . 33
2.8.2 Il ritorno dell'IA: la rivincita di Deep Blue . . . 33

3 L'avvento dell'Intelligenza Artificiale e delle reti neurali 37
3.1 La nuova era degli scacchi: l'egemonia dei motori moderni . . . 37
3.1.1 Fritz . . . 38
3.1.2 Crafty . . . 39
3.1.3 Shredder . . . 39
3.1.4 Stockfish . . . 40
3.2 La rivoluzione di AlphaZero . . . 41
3.2.1 Il gioco di un extraterrestre . . . 41
3.2.2 Caratteristiche principali del motore . . . 42
3.2.3 Il training di AlphaZero . . . 43
3.2.4 Stile e strategia di gioco . . . 44
3.3 Introduzione alle reti neurali . . . 46
3.3.1 Il neurone biologico . . . 46
3.3.2 Il percettrone . . . 47
3.3.3 Come addestrare il cervello di una macchina . . . 48
3.3.4 La funzione di attivazione sigmoidale . . . 49
3.4 Uno scontro tra titani . . . 50

4 Analisi di ChessMind Pro 53
4.1 Introduzione al ML . . . 53
4.1.1 Che cos'è il *Learning* . . . 54
4.1.2 Tipologie di problemi nel ML . . . 55
4.1.3 Uno sguardo ai diversi algoritmi . . . 56
4.2 L'addestramento di ChessMind Pro . . . 57
4.2.1 La codifica della scacchiera . . . 57
4.2.2 La preparazione del dataset . . . 59
4.2.3 Il training del modello . . . 60
4.3 Caratteristiche del motore . . . 61
4.3.1 Ambiente di sviluppo e librerie importate . . . 61
4.3.2 Rete neurale impiegata . . . 62

4.3.3 Codice sorgente 63
4.3.4 Fasi del modello neurale 65
4.4 Debolezze di ChessMind Pro 66
4.4.1 Limitazioni computazionali 66
4.4.2 Dalla forza nelle aperture alle difficoltà nei finali 66

Conclusioni **68**

Ringraziamenti **72**

A Le mosse dei pezzi **74**

B Notazione degli Scacchi **79**

Lista delle figure **81**

Lista delle tabelle **82**

Bibliografia **84**

Webgrafia **84**

Capitolo 1

Gli Scacchi oggi: tra Tradizione e Innovazione

1.1 Introduzione al gioco degli scacchi

> "Degli Scacchi è stato detto che la vita non è abbastanza lunga per essi, ma questo è il difetto della vita, non degli Scacchi." - Irving Chernev

Gli scacchi sono indubbiamente uno dei giochi più antichi e diffusi a livello globale. Attualmente, ci sono oltre 600 milioni di persone nel mondo che sanno giocare a scacchi, il che li rende il gioco da tavolo più popolare. Il numero di appassionati continua a crescere rapidamente ogni giorno: il sito chess.com ha registrato negli ultimi anni un incremento significativo dei propri iscritti, superando, al momento della stesura di questa tesi, i 170 milioni di membri. Questo numero continua a salire in modo impressionante, riflettendo l'interesse sempre crescente per i pezzi bianchi e neri.

L'antenato degli scacchi può essere considerato il *chaturanga*, un gioco da tavolo praticato in India intorno al VI secolo. Il termine "scacco", utilizzato per indicare una minaccia diretta verso il re avversario, deriva infatti dalla parola persiana *Shah*, che significa "re", e l'espressione "scacco matto", che segna invece la fine del gioco, deriva da *Shah Mat*, ovvero "il re è morto".

Nel corso degli anni hanno assunto numerose varianti e si sono evoluti attraverso varie culture, fino a diventare il gioco attuale che conosciamo oggi: 32 pezzi disposti su un campo di battaglia 8×8, nel quale i vari soldati (divisi equamente tra due contendenti) si muovono uno alla volta e secondo regole specifiche dettate dal loro rango. Ciascun pezzo è disposto a tutto, anche di dare la vita, pur di proteggere il proprio monarca, il re, il quale invece è inerme e senza poteri in confronto alla

propria consorte, la regina. La partita giunge al termine quando uno dei contendenti non è più in grado di difendere il proprio re, proclamando così lo scacco matto e sancendo la vittoria dell'avversario. A questo punto, il giocatore sconfitto ammette la propria resa alzando bandiera bianca.

1.2 Il fascino di un gioco senza tempo

Gli scacchi non sono soltanto un gioco di strategia e abilità, ma rappresentano una vera e propria sfida per la mente, la memoria e la capacità di pianificare e risolvere problemi. Un giocatore professionista di scacchi non lascia nulla al caso: deve costantemente analizzare le posizioni sulla scacchiera, prevedere le mosse future e adattare le proprie strategie in base alle mosse dell'avversario. Questo richiede un notevole livello di abilità cognitiva, flessibilità mentale e, senza dubbio, anche un certo talento.

Basti pensare che i campioni del mondo di scacchi, sia del presente che del passato, sono considerati geni indiscussi, con un QI che sfiora i 200 punti e una capacità mnemonica fuori dall'ordinario.

Gli scacchi non sono semplicemente un passatempo, ma rappresentano una battaglia mentale ardua tra due intelletti. «Gli scacchi sono una guerra sulla scacchiera. L'obiettivo è distruggere la mente dell'avversario.» ha affermato l'ex-campione del mondo statunitense Bobby Fischer. Anche Garry Kasparov lo ha definito come il "gioco più violento che esista" (in senso figurato, ovviamente). Queste affermazioni evidenziano la natura competitiva e implacabile del gioco, metafora di un intenso scontro psicologico in cui ogni mossa è finalizzata a destabilizzare l'avversario e indurlo a commettere errori. Non si gioca quindi solo con i pezzi, ma anche con la mente della persona che ci si ritrova davanti!

I più grandi strateghi della storia sono spesso stati anche abili giocatori di scacchi. La complessità del gioco richiede una profonda analisi, una pianificazione strategica e la capacità di adattarsi alle mosse dell'avversario, qualità che sono state altamente apprezzate dai più grandi pensatori e leader.

Questo gioco affascinante ha attirato una vasta gamma di persone provenienti da vari ambiti, inclusi intellettuali, matematici, filosofi, musicisti e molti altri. Persino figure illustri come Albert Einstein sono state sedotte dal fascino degli scacchi, e Dante Alighieri li ha menzionati nel suo capolavoro, la *Divina Commedia*.[1] Questo attesta la loro importanza duratura e il loro rilievo nel panorama culturale e intellettuale.

[1] Paradiso, Canto XVI, versi 136-138

In definitiva, gli scacchi sono molto più di un semplice gioco: sono una fonte di ispirazione, un campo di studio e un simbolo di ingegno e strategia che ha affascinato generazioni di persone in tutto il mondo.

1.3 Obiettivi e finalità della tesi

L'obiettivo di questa tesi è fornire un'analisi dettagliata dell'evoluzione che ha interessato i motori di scacchi nel corso della storia, a partire dalle prime implementazioni fino agli algoritmi e alle tecniche più avanzate e moderne. Il lavoro si propone di esaminare le principali tappe storiche, mettendo in luce le figure chiave che hanno maggiormente contribuito allo sviluppo dei motori scacchistici, il momento decisivo in cui il "giocatore macchina" è riuscito a superare le abilità anche dei campioni mondiali e, infine, come l'impatto rivoluzionario delle moderne tecniche di Intelligenza Artificiale abbia completamente stravolto e riformato il modo di giocare a scacchi.

Un ulteriore obiettivo è quello di presentare lo sviluppo di un motore di scacchi personale, basato su tecniche di Intelligenza Artificiale, illustrando un esempio di come una semplice rete neurale possa essere utilizzata efficacemente nei giochi. In particolare, viene dimostrato come questa rete sia in grado di giocare perfettamente una partita di scacchi, eseguendo le stesse mosse che farebbero i più grandi campioni di scacchi della storia.

Capitolo 2

L'approccio tradizionale dei motori di scacchi

2.1 I limiti della mente umana

Nel corso di una partita di scacchi, i due giocatori devono orientarsi tra l'enorme numero di possibilità di mosse che hanno a disposizione e delle potenziali risposte dell'avversario. Spesso, questo può generare ansia e stress, mettendo il cervello umano sotto una notevole pressione, aggravata dalla rilevanza della competizione e dalla tirannia del tempo che scorre inesorabilmente.[1]

Questo aspetto è ben sintetizzato nella famosa citazione di Israel Albert Horowitz: «Una mossa cattiva può rovinarne 40 ben giocate». Tale affermazione sottolinea come, nonostante la preparazione e la strategia, gli scacchi richiedano un'attenzione costante e una precisione assoluta, poiché una singola mossa sbagliata può compromettere l'esito di un'intera partita.

2.2 L'immensità dell'albero scacchistico

L'enorme complessità dell'universo scacchistico rende impossibile per i giocatori affidarsi esclusivamente a un algoritmo infallibile, costringendoli invece a fare spesso affidamento all'intuizione e alla creatività.

Secondo le stime effettuate nella sua tesi di dottorato, lo scienziato olandese Victor Allis[2] ha calcolato che il numero di partite di scacchi possibili si avvicina

[1] Durante le competizioni ufficiali, ogni giocatore ha una quantità di tempo prestabilita per completare tutte le proprie mosse; pertanto è necessario bilanciare la qualità delle proprie mosse con la gestione del tempo a disposizione.

[2] V. Allis, *Searching for solutions in games and artificial intelligence*, Ph.D. Thesis, 1994

all'ordine di grandezza di 10^{123}. Tale valore è estremamente grande e quasi incomprensibile. Per avere un'idea della sua vastità, si possono considerare i seguenti paragoni:

- il numero stimato degli atomi nell'universo osservabile è dell'ordine di 10^{80};
- con un mazzo di carte da poker, il numero totale di diverse combinazioni possibili è approssimativamente pari a 52!, che corrisponde a circa 8×10^{67}.

Se consideriamo invece il numero di posizioni diverse che possono essere effettivamente raggiunte sulla scacchiera (indipendentemente dalle mosse che sono state eseguite precedentemente), la stima si riduce a circa 10^{46}. Sarebbe possibile, quindi, immaginare l'intero universo del gioco degli scacchi come un vasto albero, dove ogni nodo rappresenta una posizione possibile sulla scacchiera e i rami (o archi) collegano i nodi tra loro, indicando le mosse legali[3] che possono essere eseguite da ciascuna posizione. Chiaramente, il primo nodo dell'albero, ovvero la radice, è il progenitore di tutti i nodi successivi; i nodi foglia, che rappresentano situazioni di scacco matto o di fine partita, si verificano quando non ci sono ulteriori mosse possibili. Fortunatamente l'albero non è uniforme, il che significa che alcune diramazioni sono più limitate di altre. Inoltre, l'albero contiene molte porzioni simili tra loro, poiché molte posizioni possono essere raggiunte attraverso diverse sequenze di mosse.

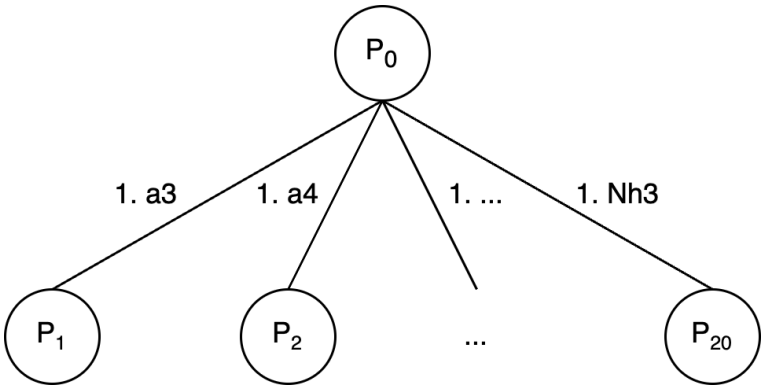


Figura 2.1: La radice comune dell'albero scacchistico

2.2.1 Il numero di Shannon

In modo molto rapido, è possibile confermare la validità dei numeri menzionati in precedenza. Immaginiamo di disporre di una scacchiera all'inizio di una partita.

[3]Una mossa legale è una mossa che rispetta tutte le regole degli scacchi. Ad esempio, è illegale non sottrarre il proprio re dalla minaccia di scacco di un pezzo avversario; pertanto, una mossa illegale è una mossa che non si può fare e, se eseguita, deve essere annullata. Nei tornei ufficiali, eseguire una mossa illegale comporta l'irrimediabile e immediata perdita della partita.

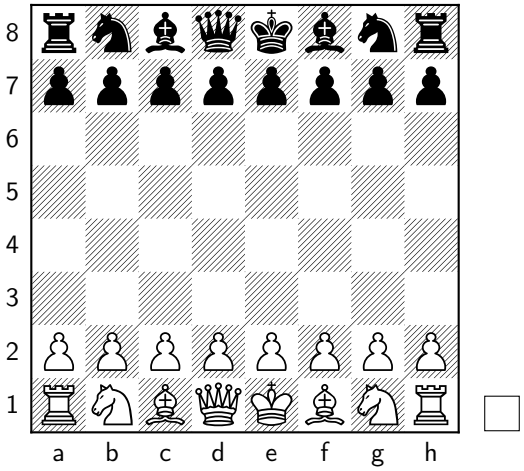


Figura 2.2: La scacchiera e la disposizione dei pezzi all'inizio di una partita di scacchi.

Il primo giocatore a muovere è il Bianco, il quale ha la possibilità di spostare uno qualsiasi dei suoi 8 pedoni di una o due caselle in avanti, così come uno dei suoi due Cavalli, che hanno due opzioni di movimento ciascuno. Questo si traduce in un totale di 20 mosse possibili per il Bianco (8 pedoni $\times$ 2 possibili mosse + 2 cavalli $\times$ 2 possibili mosse). Dopo la mossa del Bianco, è il turno del Nero. Poiché all'inizio della partita i pezzi del Nero sono disposti in modo simmetrico rispetto a quelli del Bianco, anche il secondo giocatore dispone di 20 mosse disponibili. Pertanto alla prima mossa sono già presenti $20 \times 20 = 400$ combinazioni possibili.

Dopo il primo turno di gioco, il numero di mosse legali disponibili aumenta notevolmente perché i pedoni spostati liberano il campo di azione dei pezzi posizionati sulla prima (e ottava) traversa. Eseguendo dei calcoli, è possibile giungere dopo solo quattro mosse a ben 84.998.978.956 combinazioni possibili!

numero di semimosse	numero di possibili partite
1	20
2	400
3	8902
4	197.281
5	4.865.609
6	119.060.324
7	3.195.901.860
8	84.998.978.956
...	...

Tabella 2.1: Numeri di partite possibili dopo le prime mosse di gioco.

Con qualche approssimazione e considerando che una partita di scacchi mediamente dura 40 mosse, si potrebbe stimare il numero totale di partite teoricamente giocabili:

$$\underbrace{10^3 \times 10^3 \times \cdots \times 10^3}_{40 \text{ volte}} = \left[10^3\right]^{40} = 10^{120} \text{ partite totali} \tag{2.1}$$

Questa approssimazione appena ottenuta prende il nome di *numero di Shannon*,[4] ed è un indicatore fondamentale della complessità di questo gioco.

La capacità di un giocatore di alto livello va ben oltre il "trovare il matto in uno",[5] o "trovare l'attacco doppio"[6] o ancora "posizionare il cavallo in una casella strategica" (o comunque qualsiasi combinazione lunga 1 o massimo 2 mosse). Ciò che può fare davvero la differenza è la capacità di ricordare lunghe linee di apertura (anche oltre 20 mosse dopo l'inizio della partita), nonché gli innumerevoli pattern di scacco matto e le strategie per gestire ogni posizione a proprio vantaggio. Questi dettagli possono rivelarsi decisivi per l'esito finale di una partita di scacchi.

Nonostante gli sforzi di memorizzazione e l'esperienza accumulata nel corso degli anni, il vasto numero di possibilità rimane una sfida costante. Inoltre, va considerato che la mente umana non è perfetta e può essere influenzata da numerosi fattori, sia interni che esterni alla scacchiera. La stanchezza, lo stress e la pressione del momento possono avere un impatto significativo sulle capacità cognitive e decisionali di un giocatore, anche di quelli più esperti.

2.2.2 Lo "scacchista camionista"

Secondo Luigi De Rosa, un importante e storico istruttore di scacchi del panorama salentino, «per eccellere nel mondo degli scacchi, è necessario vestire i panni di un "esperto camionista", piuttosto che quelli di un abile stratega o tattico». La frase, seppur provocatoria, esprime perfettamente il concetto che vi è alla base: così come un veterano camionista conosce ogni strada e vicolo della rete stradale italiana, un grande giocatore di scacchi deve essere in grado di muoversi agilmente nel complesso labirinto delle possibili mosse e contro-mosse che scaturiscono da una posizione sulla

[4] Conosciuto anche come *numero di Shannon-Weaver* o *Shannon game-tree complexity*, prende il nome dal matematico e ingegnere dell'informazione Claude Shannon (1916-2001), che lo introdusse nel suo articolo intitolato *Programming a Computer for Playing Chess* pubblicato sulla *Philosophical Magazine* nel 1950.

[5] Il termine "matto", oppure "scacco matto", indica una situazione del gioco degli scacchi in cui il re di uno dei due giocatori è sotto attacco (o "sotto scacco") e non può sfuggire a questa minaccia. Per "matto in uno" si riferisce ad una determinata situazione in cui un giocatore può dare scacco matto all'avversario con una singola mossa.

[6] Un attacco doppio è una mossa con la quale un proprio pezzo minaccia simultaneamente due pezzi dell'avversario non opportunamente difesi, rendendo probabile la cattura di almeno uno di essi.

scacchiera. In questo contesto, la memoria gioca un ruolo importante, poiché ricordare le varie configurazioni e i vari schemi presenti nell'immenso albero dei pezzi bianchi e neri si rivela sicuramente di grande aiuto. Con l'avanzare dell'età, diventa sempre più difficile mantenere questa vasta conoscenza, rendendo può complesso competere ai massimi livelli degli scacchi. Di conseguenza, la capacità mnemonica supera in importanza le abilità tattiche a breve termine, poiché rappresenta la chiave per anticipare e pianificare strategie a lungo raggio durante una partita.

2.3 L'avvento dei motori scacchistici

L'avvento dei motori scacchistici ha segnato un punto di svolta nel mondo degli scacchi, trasformando un gioco millenario in un campo di battaglia per l'intelligenza artificiale e l'ingegno umano.

2.3.1 I motori di scacchi nascono con l'informatica

Fin dagli anni '40 del secolo scorso, l'idea di automatizzare i processi di calcolo per migliorare le capacità di gioco degli scacchi ha iniziato a prendere forma. La complessità e la lunghezza dei calcoli necessari per eseguire mosse ottimali sulla scacchiera hanno sollevato la necessità di sistemi automatici che potessero migliorare la velocità e l'accuratezza del gioco.

Parallelamente, l'informatica stessa stava emergendo come una nuova e rivoluzionaria disciplina. La nascita dei primi computer, come l'ENIAC[7] negli Stati Uniti, e il lavoro pionieristico di figure come Alan Turing hanno gettato le basi per la programmazione automatica e l'elaborazione elettronica dei dati. Con le sue capacità di calcolo avanzate e la possibilità di implementare algoritmi complessi, l'informatica ha giocato un ruolo fondamentale nello sviluppo dei primi motori scacchistici. È quindi possibile affermare che l'informatica e i motori di scacchi sono nati e si sono evoluti parallelamente.

Questi sviluppi si sono svolti in concomitanza con l'emergere di nuove discipline come la teoria dell'informazione, la cibernetica, la teoria dei giochi e la teoria della probabilità, le quali hanno iniziato a prendere voga nello stesso periodo. Gli scacchi sono stati fortunati ad avere numerosi grandi matematici e informatici che, oltre a eccellere nelle loro discipline, si sono dedicati con passione anche allo sviluppo dei motori scacchistici.

[7] L'ENIAC (*Electronic Numerical Integrator and Computer*) fu il primo calcolatore interamente elettronico costruito nel 1946; pesava 30 tonnellate e occupava uno spazio di 140 metri quadrati.

2.3.2 Cenni sulla teoria dei giochi

John von Neumann (1903-1957) fu una delle personalità più importanti del secolo scorso e uno dei primi ad approfondire i concetti di gioco a *informazione perfetta* e a *somma zero*, applicandoli agli scacchi. Ogni posizione sulla scacchiera, infatti, deriva da mosse precise e ben calcolate, non influenzate da eventi casuali o probabilistici (come il lancio di una moneta). Inoltre, in un gioco a somma zero, qualsiasi vantaggio o svantaggio di un giocatore corrisponde esattamente a uno svantaggio o vantaggio per l'avversario.

Il grande matematico approfondì inoltre il concetto di *induzione a ritroso*[8], una tecnica che permette di risolvere un problema partendo dall'ultima azione eseguita fino alle mosse iniziali, rivelando la strategia ottimale per navigare all'indietro nel complesso albero decisionale degli scacchi (vedi Paragrafo 2.2). Nello stesso lavoro pionieristico sulla teoria dei giochi, von Neumann introdusse il *teorema del minimax*, secondo il quale la scelta ottimale di un giocatore mira a minimizzare le massime perdite possibili e a massimizzare il suo potenziale vantaggio (vedi Paragrafo 2.4.3).

In tale campo, assume grande rilevanza anche il teorema di Zermelo-Kuhn, coniato dai matematici Ernst Zermelo (1871-1953) e Harold Kuhn (1925-2014). Esso stabilisce che in un gioco finito ad informazione perfetta, tra due giocatori che muovono alternativamente e dove il caso non influisce sul processo decisionale, almeno uno dei giocatori ha una strategia vincente. Inoltre, una qualsiasi posizione di scacchi deve necessariamente rientrare in una delle seguenti condizioni:

1. la posizione è vincente per il giocatore Bianco
2. la posizione è vincente per il giocatore Nero
3. la posizione è patta,[9] ossia se entrambi giocano correttamente, è possibile forzare la parità.

Queste formulazioni teoriche, sebbene ancora astratte, sono ben presto destinate a trovare le proprie applicazioni pratiche, come vedremo nei prossimi paragrafi.

[8] John Von Neumann, Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[9] Una partita a scacchi può terminare anche con una situazione di parità, nota come patta, in cui nessuno dei due giocatori ottiene la vittoria. La patta può essere raggiunta in diversi modi: quando il giocatore che deve muovere non ha più mosse legali a disposizione e non ha il proprio re sotto scacco, quando una stessa posizione si ripete per tre volte durante la partita, oppure se dopo cinquanta mosse consecutive non viene effettuata né una cattura di pezzo né una mossa di pedone. Una partita può concludersi in parità anche dopo comune accordo tra i due giocatori.

2.3.3 Turochamp: il primo motore della storia

Senza dubbio, uno dei grandi matematici che ha tradotto questi concetti teorici in pratica è stato il teorico e crittografo britannico Alan Turing (1912-1954), oggi considerato uno dei padri dell'informatica moderna.

Durante la Seconda Guerra Mondiale, Alan Turing ha svolto un ruolo fondamentale presso Bletchley Park, il centro governativo britannico per la crittoanalisi. Qui ha guidato gli sforzi per decodificare il codice ENIGMA utilizzato dai tedeschi, inventando la celebre macchina elettromeccanica chiamata *Bombe*. Questo dispositivo ha rivoluzionato il processo di decrittazione dei messaggi cifrati, apportando un contributo decisivo agli sforzi bellici degli Alleati.

Nel 1936 ha introdotto il concetto rivoluzionario di *macchina di Turing*,[10] una descrizione logica teorica di un calcolatore in grado di manipolare simboli su un nastro di lunghezza infinita secondo una serie di istruzioni ben definite. Nel 1950, ha ulteriormente influenzato il campo dell'intelligenza artificiale con il suo "test di Turing",[11] ossia un test che si proponeva di valutare se una macchina fosse in grado di esibire comportamenti intelligenti indistinguibili da quelli umani.

Dal punto di vista scacchistico, Alan Turing riuscì a sviluppare nel 1948 uno dei primi programmi scacchistici con l'aiuto del suo amico e collega David Gawen Champernowne (1912-2000). Questo programma, noto con il nome di *Turochamp*, era estremamente rudimentale e limitato dalle tecnologie dell'epoca, tanto che l'algoritmo fu concepito su carta in assenza di una macchina disponibile per la programmazione. Durante le partite, Turochamp impiegava circa mezz'ora per elaborare ogni mossa, a causa della necessità di eseguire in modo rigido una serie di lunghe operazioni.

È importante sottolineare che Turing non aveva l'obiettivo di sviluppare un programma in grado di giocare perfettamente a scacchi o di battere il campione del mondo. Piuttosto, il suo intento era tradurre il proprio modo di pensare e giocare a scacchi in istruzioni facilmente eseguibili in modo automatico.

Nonostante l'algoritmo originale sia andato perduto nel corso del tempo, è stato recentemente ricostruito grazie alle molte testimonianze storiche. È stato inoltre implementato un vero e proprio programma scacchistico[12] che oggi ci permette di giocare contro il computer di Turing (per giocare a Turochamp, è possibile visitare il sito mdoege.github.io/nimTUROCHAMP).

[10] Alan Turing, *On Computable Numbers, with an Application to the Entscheidungsproblem.* 1936

[11] Alan Turing, *Computing Machinery and Intelligence*, Mind, vol. 59, no. 236, 1950, pp. 433-460.

[12] Per visionare il codice, visitare il seguente file prelevato dal repository github pyturochamp.py.

Nel 2012, il "rinato" Turochamp è stato impiegato in una celebrazione per il centenario della nascita di Turing, con il grande campione del mondo Garry Kasparov chiamato a sfidarne le capacità. A distanza di sessant'anni dalla sua prima apparizione nel panorama internazionale, Turochamp, purtroppo, non è riuscito a mostrarsi all'altezza. Difatti, la partita si è conclusa in una breve miniatura[13] e sono bastate solo sedici mosse a Kasparov per infliggergli lo scacco matto. Questa sfida ha messo in luce le significative difficoltà che i motori di scacchi hanno affrontato nel tentativo di affermarsi a livello globale. Tuttavia, questo è solo l'inizio di un'evoluzione destinata a rivoluzionare radicalmente il panorama, come analizzeremo nel corso di questa tesi.

Esaminiamo di seguito le principali regole del primo motore di scacchi della storia.

Valutazione di una posizione

Turing ha introdotto un numero limitato di regole, semplici e facilmente convertibili in istruzioni per una macchina, ma sufficientemente efficaci per permettere al suo computer di valutare una qualsiasi posizione di scacchi. In particolare, Turing ha introdotto l'utilizzo di parametri numerici, la cui somma totale determinava quale giocatore fosse in vantaggio: un punteggio positivo indicava il vantaggio del Bianco, mentre un punteggio negativo segnalava il vantaggio del Nero (ovviamente un valore pari a 0 stava ad indicare la perfetta parità tra i due giocatori). I singoli punteggi venivano assegnati in base ai seguenti criteri:

- Pezzi presenti sulla scacchiera: ogni pezzo aveva un suo valore numerico; Turing ha attribuito i seguenti punteggi: Pedone = 1, Cavallo = 3, Alfiere = 3.5, Torre = 5, Regina = 10.

- Mobilità dei pezzi (esclusi pedoni e re): veniva aggiunta la radice quadrata del numero di mosse che ogni pezzo poteva eseguire (contando la cattura di un pezzo come mossa doppia).

- Sicurezza dei pezzi: veniva aggiunto 1 punto ad ogni pezzo che era difeso una volta e 1.5 punti ad ogni pezzo difeso due volte.

- Sicurezza del re: veniva tolto un numero di punti variabile a seconda delle case[14] "scoperte" davanti al proprio re.

[13] Una miniatura è una partita breve della durata di massimo 25 mosse. Di solito, la partita termina così in fretta per via di uno o più errori fatali commessi nelle prime mosse d'apertura.

[14] Il termine "casa", equivalente al termine "casella", si riferisce a ciascuno dei 64 quadrati che compongono la scacchiera e su cui i pezzi possono muoversi.

- Arrocco:[15] veniva aggiunto 1 punto ogni qualvolta che l'arrocco era possibile oppure era stato appena eseguito.
- Pedoni: venivano attribuiti 0.2 punti ogni volta che un pedone avanzava di una casa e 0.3 punti per ogni pedone difeso da uno o più pezzi.
- Scacco: veniva attribuito 1 punto ad ogni minaccia di scacco matto e 0.5 punti ad ogni scacco eseguito.

"Posizione aperta" o "Posizione morta"

Una posizione è definita "aperta" quando molti pezzi sono in contatto con quelli dell'avversario, rendendo possibili numerosi scambi, catture e minacce di scacco matto. È fondamentale che il motore possa determinare facilmente se una posizione richiede un'attenta analisi di tutte le varianti derivanti dai vari scambi, oppure una semplice valutazione statica della posizione, cosa che avviene nelle posizioni chiuse (o "morte").

Mosse da analizzare

In base ai criteri appena esplicitati, il motore di Turing doveva esaminare tutte le mosse legali di un giocatore che scaturivano da una determinata posizione, seguite da ogni possibile risposta dell'avversario. La complessità di questa analisi variava in funzione del *ply* (livello) impostato, dove il *ply* indica il numero di semimosse consecutive che il motore doveva calcolare, e quindi la profondità della sua analisi.

2.4 Il contributo del padre della teoria dell'informazione

Tra i pionieri fondamentali nel campo dell'informatica e della teoria dell'informazione, non può mancare Claude Shannon (1916-2001) nell'elenco dei grandi fautori dei motori scacchistici.

Claude Shannon è celebre per aver introdotto il concetto di *bit*, l'unità fondamentale di informazione che ha rivoluzionato la quantificazione e la manipolazione dei dati in modo preciso e sistematico. In collaborazione con il fisico Harry Nyquist,

[15] L'arrocco è una mossa speciale, praticabile una sola volta nel corso di una partita, che permette di muovere contemporaneamente il proprio re e una delle due torri. È fondamentale eseguire l'arrocco per aumentare la sicurezza del proprio monarca e mettere velocemente in gioco la torre isolata nell'angolo.

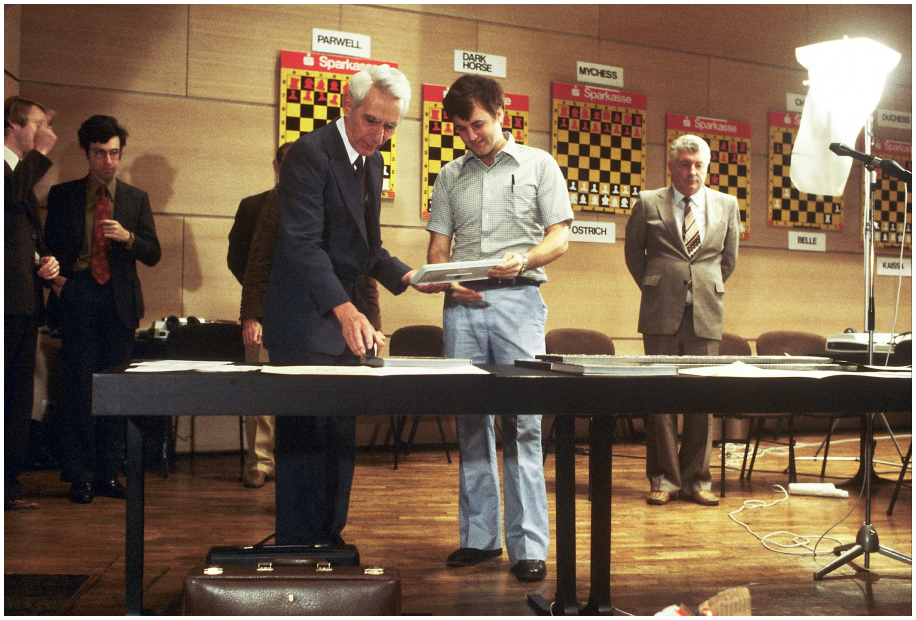


Figura 2.3: Claude Shannon durante il 3° Campionato del Mondo di Scacchi per Computer, Festival Ars Electronica, 1980. Fonte: www.flickr.com.

ha formulato il Teorema del Campionamento Ideale (1949), un pilastro della teoria dei segnali e della telecomunicazione moderna. I suoi contributi nella teoria dei codici e nella compressione dei dati hanno avuto un impatto significativo su numerosi settori. Nel contesto degli scacchi, le teorie di Shannon hanno fornito fondamenta essenziali per lo sviluppo dei motori scacchistici, evidenziando la loro importanza attraverso la profonda passione di Shannon per questo gioco e la capacità di testare le sue innovazioni informatiche in questo campo.

In una biografia di Shannon[16] sono riportati gli incontri tra lui e Turing, avvenuti a partire dal 1943, durante la Seconda Guerra Mondiale, quando i due erano impiegati come crittografi rispettivamente dal governo britannico e statunitense. Durante i loro colloqui informali, ebbero modo di scambiarsi molti pareri riguardo la meccanizzazione del pensiero umano, la progettazione di macchine intelligenti e, perché no . . . anche il gioco degli scacchi!

2.4.1 La formula di Shannon

Si deve a Shannon l'approfondimento e il miglioramento delle regole di Turochamp, in particolare l'introduzione di nuove variabili per la valutazione delle posizioni sulla scacchiera. A tal proposito ha introdotto una formula $f(P)$ che, in base alla dispo-

[16] Jimmy Soni, Rob Goodman, *A Mind at Play: How Claude Shannon Invented the Information Age*. Simon & Schuster, 2017.

sizione dei pezzi, fornisce un indice numerico che determina quale giocatore è in vantaggio. Tale funzione doveva tener conto di alcune informazioni, quali:

- Il numero dei pezzi presenti sulla scacchiera per ogni giocatore; similmente a quanto non abbia già fatto l'amico Turing, Shannon ha attribuito un punteggio numerico ad ogni pezzo: $K = 200$, $Q = 9$, $R = 5$, $B = 3$, $N = 3$, $P = 1$.[17] Nonostante la sua limitata capacità di movimento, al Re è stato attribuito un valore inestimabile, poiché il gioco non potrebbe proseguire senza di lui.

- La struttura pedonale: il giocatore che possiede dei pedoni doppiati, arretrati e/o isolati ha sicuramente un leggero svantaggio.[18]

- Sicurezza del re (soprattutto prima del finale).[19]

Applicando questi principi, è stato in grado di elaborare un semplice criterio di valutazione, riassumibile nella seguente formula:

$$\begin{aligned} f(P) \;=\; & 200 \cdot (K - K') + 9 \cdot (Q - Q') + 5 \cdot (R - R') + \\ & + 3 \cdot (B - B' + N - N') + (P - P') + \\ & - \frac{1}{2} \cdot (D - D' + S - S' + I - I') + \frac{1}{10} \cdot (M - M') \end{aligned} \tag{2.2}$$

dove:

- le lettere senza apostrofo si riferiscono ai pezzi del Bianco, mentre le lettere con l'apostrofo si riferiscono ai pezzi del Nero

- i pezzi sono stati indicati utilizzando la notazione PGN (*Portable Game Notation*, una delle più diffuse ad oggi per rappresentare una partita di scacchi anche in formato digitale)[20]

[17] L'abbreviazione dei vari pezzi degli scacchi usando le loro iniziali in inglese è tipica in molte notazioni: K per il Re (King), Q per la Regina (Queen), R per la Torre (Rook), B per l'Alfiere (Bishop), N per il Cavallo (Knight), P per il Pedone (Pawn).

[18] Due pedoni sono "doppiati" quando si trovano sulla stessa colonna; un pedone si definisce "arretrato" quando si trova dietro rispetto ai pedoni dello stesso colore posti sulle colonne adiacenti; infine un pedone è "isolato" quando non ha pedoni dello stesso colore sulle colonne adiacenti. Tutte queste situazioni rendono i pedoni deboli, poiché non possono essere protetti da altri pedoni dello stesso schieramento.

[19] Una partita a scacchi si suddivide generalmente in tre fasi distintive: l'apertura, il mediogioco e il finale. Il finale rappresenta la fase avanzata della partita, caratterizzata dalla presenza di pochi pezzi sulla scacchiera. L'obiettivo principale in questa fase è ottenere un vantaggio decisivo con i pezzi rimasti, ad esempio attivando il proprio Re e/o promuovendo un proprio pedone.

[20] Vedi nota 17 e appendice B.

- i simboli D, S, I stanno ad indicare rispettivamente il numero di pedoni doppiati, arretrati e isolati (tali termini sono preceduti da un segno meno proprio perché rappresentano uno svantaggio)
- il simbolo M è un indice della mobilità dei pezzi, ottenuta per esempio contando il numero di mosse legali a disposizione di ogni giocatore

È importante specificare che i coefficienti $-\frac{1}{2}$ e $\frac{1}{10}$ sono mere stime approssimative, non basate su alcuna verità o principio scacchistico consolidato.

2.4.2 I difetti della valutazione

Nonostante il lavoro di Shannon fosse meticoloso e ben organizzato, non considerava tutti i possibili fattori che possono determinare un reale vantaggio per uno dei giocatori. Questo era comprensibile, dato che Shannon non era un esperto scacchista e non dedicava tutto il suo tempo a questo gioco. Tali intuizioni potevano venire solo a un vero professionista del settore, non a un semplice appassionato.

Infatti, ci sono molti altri concetti che possono effettivamente costituire un vantaggio decisivo e che non sono inclusi nella formula di Shannon. Ad esempio, posizionare una torre su una colonna aperta è uno dei principi fondamentali del gioco, noto anche ai giocatori meno esperti. In generale, un giocatore ha una posizione migliore se riesce a collocare i propri pezzi in caselle dove hanno massima mobilità e il miglior raggio d'azione possibile. Questo principio si applica anche agli alfieri e ai cavalli. Un altro fattore importante è il *valore relativo* dei pezzi: un cavallo non vale sempre 3 punti, così come un alfiere. Il loro valore dipende dalla posizione che occupano. In alcune situazioni, un pedone può essere più potente di una torre, oppure può essere preferibile avere un alfiere piuttosto che un cavallo (o una torre).

È interessante anche considerare in una partita di scacchi eventuali sacrifici, ossia delle mosse in cui un giocatore cede volontariamente un pezzo di valore importante per ottenere un vantaggio strategico o tattico nel corso della partita. È un modo per adescare i pezzi avversari in delle caselle scomode dove hanno poche possibilità di manovra, oppure può essere utile per aprire linee d'attacco verso il re avversario. Il sacrificio è una mossa spettacolare e geniale, ma può anche rivelarsi un errore fatale. La chiave del successo risiede nella bravura, nella coordinazione e nella velocità dei pochi pezzi rimasti, che devono sfruttare al meglio la loro posizione per scardinare le difese dell'avversario, dare scacco matto, oppure riguadagnare il materiale sacrificato con gli interessi.

Di seguito è riportato un esempio di valutazione di una posizione utilizzando la formula di Shannon confrontata con la valutazione di un motore scacchistico

moderno. Analizzeremo e confronteremo i risultati ottenuti dai due motori. La partita in questione è stata giocata durante la prima edizione dell'*American Chess Congress* a New York nel 1857, tra il futuro campione del mondo Paul Morphy (1837-1884) e l'abile scacchista Louis Paulsen (1833-1891).[21]

La Figura 2.4 mostra la disposizione dei pezzi durante la suddetta partita alla 17^a mossa. È il turno di Morphy (il giocatore Nero), il quale ha impiegato ben 12 minuti per decidere la continuazione migliore.

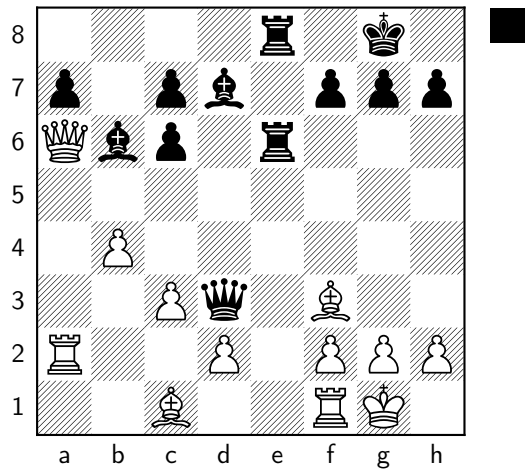


Figura 2.4: Posizione sulla scacchiera alla 17^a mossa della partita tra Louis Paulsen e Paul Morphy all'American Chess Congress (1857).

La posizione sembra apparentemente in equilibrio, con i pezzi bianchi e neri che si equivalgono in numero. Infatti, prendendo in considerazione la formula 2.2 menzionata in precedenza, questa restituisce un valore di $+0.7$, indicando quindi un leggero vantaggio per Paulsen, dovuto probabilmente alla migliore disposizione dei pedoni in confronto di quella dell'avversario. Tuttavia, è interessante notare che, analizzando la medesima posizione con il motore Stickfish 16 NNUE (uno dei più utilizzati al giorno d'oggi), la valutazione è invece di -4.0.[22] Senza dubbio una netta differenza!

È ancora più stupefacente la continuazione: Morphy infatti ha giocato la brillante e spettacolare mossa **17... ♕×f3!**[23] (Figura 2.5), un vero e proprio sacrificio di donna![24] Questa è una mossa rara e rischiosa, dato che si è volontariamente "get-

[21] Per visionare la partita completa, visitare il sito www.chessgames.com.

[22] La valutazione di un motore di scacchi è un valore numerico che indica la relativa forza della posizione attuale sulla scacchiera. In genere, questa valutazione è approssimabile al vantaggio materiale che avrebbe uno dei due giocatori, equivalentemente al valore di un pedone.

[23] Questa mossa indica che il Nero ha catturato un pezzo nella casella f3 con la propria donna; il punto esclamativo è utilizzato per indicare una "mossa geniale". Per comprendere meglio la notazione, vedere Appendice B.

[24] La "donna" è un termine molto comune negli scacchi per indicare la regina.

tato via" il pezzo più potente del proprio schieramento in cambio di un probabile vantaggio strategico a lungo termine.

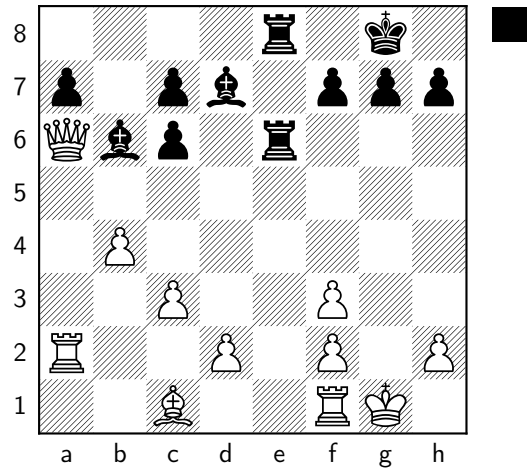


Figura 2.5: Paulsen - Moprhy; posizione sulla scacchiera dopo il sacrificio di donna di Morphy e la conseguente ricattura della regina da parte di Paulsen con il proprio pedone nella casella f3.

Riposizionando questa posizione nei nostri motori, Stockfish continua a confermare la sua analisi, stabilendo una valutazione di -3.8 e indicando nuovamente un notevole vantaggio per il Nero. Tuttavia, la vera sorpresa arriva dall'analisi del motore di Shannon, il quale considera il sacrificio di donna un grave errore, portando la valutazione a $\ldots +5.5$!

Dai dati presentati emerge chiaramente che la formula di Shannon è lontana dall'essere precisa e realistica. Questo dimostra che valutare una posizione basandosi principalmente sul potere dei pezzi disponibili, anziché considerare effettivamente l'efficacia e l'utilità di tali pezzi per il raggiungimento dell'obiettivo finale, è impossibile.

Il sacrificio di Morphy ha praticamente sguarnito tutte le difese del re avversario, non lasciando alcun pezzo nei suoi paraggi pronto ad accorrere in suo soccorso. Nonostante il Bianco abbia una donna di vantaggio, egli non è in grado di sfruttarla a dovere, essendo la regina posizionata lontano dal centro degli attacchi del Nero e non rappresentando alcuna minaccia immediata per lo schieramento avversario.

In sole 10 mosse, Morphy è riuscito a capitalizzare il suo vantaggio di sviluppo e la migliore attività dei suoi pezzi. Paulsen, per difendersi da una minaccia di scacco matto, ha dovuto sacrificare la regina e una torre per evitare danni maggiori, perdendo così troppo materiale e, di conseguenza, la partita. A titolo informativo,

Stockfish ha valutato la posizione in Figura 2.6 a -11.1, un chiaro segno di una vittoria schiacciante per il Nero, a meno di errori clamorosi!

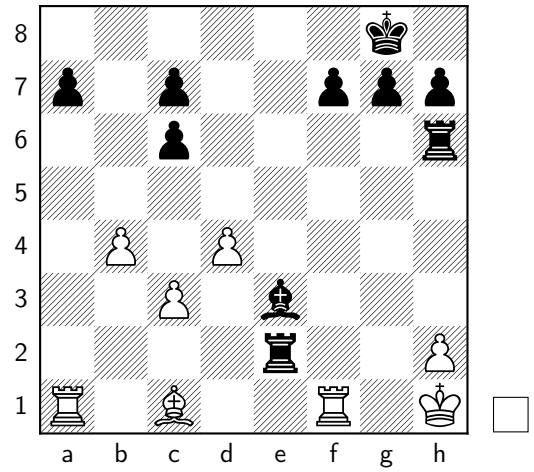


Figura 2.6: Paulsen - Moprhy, posizione sulla scacchiera alla 28^a mossa. Il Bianco ha già ceduto la Regina, e ora è costretto a sacrificare la torre in f1 per poter difendere la minaccia di matto.

2.4.3 La strategia A

Abbiamo esaminato come Claude Shannon abbia sviluppato un metodo per valutare la disposizione dei pezzi sulla scacchiera. Ora esploreremo come abbia creato un algoritmo per determinare quale mossa preferire rispetto ad un'altra, noto come la strategia "metodo A" dell'albero di Shannon.

Definizione formale

Consideriamo una certa posizione P e supponiamo che sia il turno del Bianco (o del Nero). Indichiamo con N il numero totale di mosse possibili a disposizione del Bianco e chiamiamo con $m_1, m_2, \ldots, m_N$ le mosse legali che derivano da P. Supponiamo inoltre di avere a disposizione la formula di Shannon $f(P)$ analizzata in precedenza ad una profondità iniziale di 1 ply. La mossa m_k da scegliere è quella che massimizza il valore della funzione, ossia la mossa per cui $f(P_k)$ è massima.

Questo algoritmo si basa su una ricerca in profondità dell'albero delle possibili mosse a partire dalla posizione attuale sulla scacchiera. L'idea principale è quella di esplorare tutte le possibili sequenze di mosse fino a una certa profondità (che abbiamo indicato con ply) e valutare le posizioni raggiunte alla fine di ciascuna sequenza. La

valutazione avviene utilizzando la funzione di valutazione precedentemente discussa (formula 2.2).

La strategia per la scelta della mossa si basa sulla ricerca di varianti che minimizzino il rischio massimo (la cosiddetta *ricerca minimax*). Ciò significa che, per ciascun livello di profondità della ricerca, l'algoritmo cerca di (supponendo sempre che sia il turno del Bianco):

1. Massimizzare il proprio vantaggio: il Bianco sceglie la mossa che gli fornisce il miglior risultato possibile.

2. Minimizzare il vantaggio del Nero: si suppone che il secondo giocatore giochi alla perfezione e che, tra tutte le possibili mosse di risposta, scelga sempre quella migliore.

Esempio pratico

Supponiamo di trovarci in una determinata posizione P_0, che identifichiamo con il nodo A. Per semplicità, ipotizziamo che il Bianco abbia a disposizione solo due mosse legali, che chiamiamo b e c. Di conseguenza, il nodo A avrà due nodi figli, che chiameremo B e C, e corrispondenti rispettivamente alle posizioni risultanti dalle mosse b e c del Bianco. Dopo la mossa del Bianco, anche il Nero avrà due mosse legali disponibili per ogni mossa del Bianco. Ciò significa che ciascuno dei nodi B e C avrà a sua volta due nodi figli. Chiamiamo questi nodi D ed E per i figli di B, e F e G per i figli di C, e indichiamo con le lettere minuscole d, e, f, g le mosse da eseguire per raggiungere tali nodi (vedi Figura 2.7). Abbiamo creato un albero fino a due livelli di profondità (2 ply).

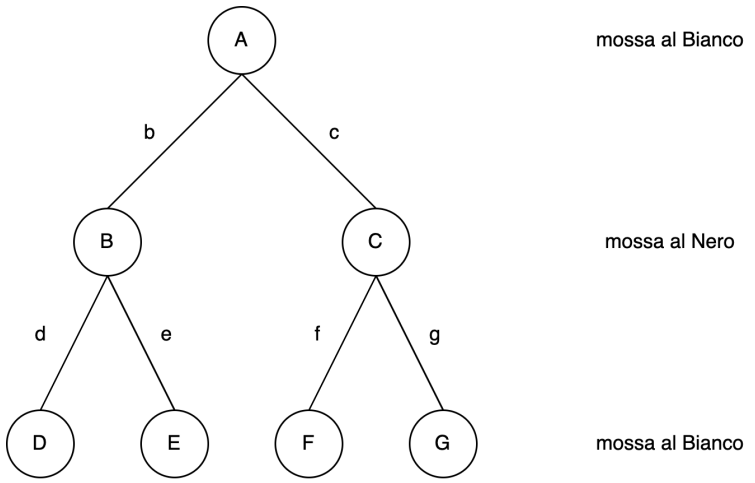


Figura 2.7: Esempio di applicazione della Strategia A su un albero decisionale.

Giunti a questo punto, si esaminano tutte le posizioni terminali (le foglie) dell'albero utilizzando la funzione di valutazione. Supponiamo i seguenti valori di fantasia:

$$f(D) = 3 \qquad f(E) = -5 \qquad f(F) = 4 \qquad f(G) = 0$$

Adesso, a seconda di chi spetta la mossa, si eseguono due algoritmi differenti:

- turno del Nero (minimizzazione): bisogna scegliere la mossa per cui la funzione restituisce il valore minimo; nel nostro caso:
  - per il nodo B: $\min(D, E) = -5$, va scelta la mossa e
  - per il nodo C: $\min(F, G) = 0$, va scelta la mossa g
- turno del Bianco (massimizzazione): è necessario scegliere la mossa migliore, quindi il valore massimo:
  - per il nodo A: $\max(B, C) = 0$, va scelta la mossa c

In conclusione, il Bianco dovrebbe scegliere la mossa c che conduce alla posizione del nodo C, garantendosi il maggior vantaggio possibile (di minimo 0 punti), supponendo che il suo avversario giochi in modo perfetto.

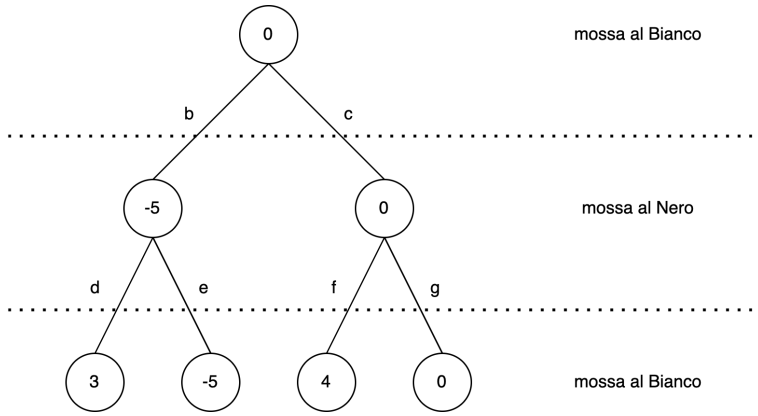


Figura 2.8: Continuazione dell'esempio precedente: le valutazioni delle posizioni nei nodi terminali si sono propagate ai nodi genitori seguendo i metodi illustrati.

2.4.4 La strategia B

La Strategia A ha contribuito significativamente allo sviluppo degli algoritmi per il gioco degli scacchi. Tuttavia, questa strategia presentava alcune limitazioni importanti:

- Complessità: basandosi su una ricerca esaustiva delle possibili mosse, la Strategia A affrontava una crescita esponenziale della complessità computazionale al crescere del numero del ply (ossia delle mosse da analizzare); questo rappresentava una sfida considerevole dato soprattutto il limitato potenziale computazionale dell'epoca.

- Disuguaglianza delle posizioni: non tutte le posizioni di gioco sono uguali. Alcune richiedono un'analisi dettagliata delle sequenze tattiche che coinvolgono molti scambi di pezzi, mentre altre possono essere risolte con una singola mossa decisiva tra molte alternative. Questa variabilità rendeva poco efficiente un'analisi uniforme e completa di tutte le casistiche possibili.

Per superare queste difficoltà, è emersa la Strategia B, con l'obiettivo di migliorare i seguenti concetti chiave:

- Nuove funzioni euristiche di valutazione: queste funzioni sono progettate per esplorare il più possibile in profondità, valutando l'esito di una posizione soltanto una volta aver raggiunto delle 'posizioni morte'. Questo metodo utilizza una profondità variabile anziché fissa e include una ulteriore funzione per la valutazione della stabilità della posizione stessa.

- Approcci di ricerca selettiva: invece di esplorare tutte le possibili mosse, la Strategia B ha implementato tecniche come l'*alpha-beta pruning* (vedi Paragrafo 2.5) per ridurre il numero di mosse da considerare. Questo consente di concentrare le risorse computazionali sulle mosse più promettenti, migliorando l'efficienza complessiva dell'algoritmo.

Per Shannon, perfezionare queste funzioni euristiche rappresentava una sfida non di poco conto. Era necessario il contributo di esperti programmatori per migliorare sia la sua funzione, ancora poco accurata, che per introdurre nuove funzioni in grado di considerare gli altri fattori precedentemente considerati.

2.5 La potatura alfa-beta

John McCarthy (1927-2011) è stato uno dei precursori nell'applicazione dell'intelligenza artificiale ai giochi strategici. È noto soprattutto per il suo lavoro sull'algoritmo minimax e sulla *potatura alfa-beta* (in inglese *alfa-beta pruning*). Questa tecnica è un'ottimizzazione dell'algoritmo minimax (già analizzato con Shannon) che riduce significativamente il numero di nodi dell'albero di ricerca da esaminare, eliminando la necessità di valutare la posizione di ogni nodo foglia. Come dice il nome stesso,

la potatura alfa-beta funziona utilizzando due parametri, α e β, per tagliare i rami dell'albero di che non influenzano la scelta finale. Supponiamo di disporre di una posizione di scacchi con mossa al Bianco, allora:

- α sta a rappresentare il punteggio minimo che il primo giocatore (il Bianco) può raggiungere; all'inizio viene posto a $-\infty$;
- β sta a rappresentare il punteggio massimo che il secondo giocatore (il Nero) può raggiungere; all'inizio viene posto a $+\infty$.

Nella ricerca minimax con potatura alfa-beta, durante l'esplorazione dell'albero di gioco, vengono aggiornati continuamente i valori di α e β per ciascun nodo. Questi valori rappresentano rispettivamente il miglior valore massimo garantito per il Bianco e il miglior valore minimo garantito per il Nero fino a quel punto nella ricerca. Durante l'esplorazione dell'albero:

1. se per un dato nodo α diventa maggiore β, vuol dire che il Bianco ha già trovato un'ottima mossa, indipendentemente dalla risposta dell'avversario
2. a tal punto, non è necessario continuare l'esplorazione dell'albero in corrispondenza di quel nodo, dato che il Nero non sceglierà mai una mossa inferiore che permette al Bianco di ottenere un grande vantaggio
3. pertanto, ogni albero associato a questo nodo con $\alpha > \beta$ viene 'potato', cioè rimosso dalla ricerca dell'algoritmo.

In sostanza, la funzione di valutazione $f(P)$ deve necessariamente soddisfare la condizione:

$$\alpha \leq f(P) \leq \beta \tag{2.3}$$

2.5.1 Complessità computazionale

Definiamo con b il *fattore di branching*, che rappresenta il numero medio di mosse legali disponibili per ogni posizione, e con d la profondità dell'albero di gioco esplorato (ply). Nel metodo minimax senza potatura alfa-beta, la complessità computazionale è $O(b^d)$ Questo perché ogni nodo fino alla profondità d deve essere valutato completamente.

D'altra parte, nel metodo con potatura alfa-beta, la complessità può variare da un caso peggiore di $O(b^d)$ (identico al metodo senza potatura alfa-beta) a un caso migliore di $O(b^{\frac{d}{2}})$. Quest'ultimo rappresenta un notevole miglioramento, sebbene dipenda dall'efficacia con cui vengono scelti i primi nodi da esplorare. Nonostante la

complessità possa avvicinarsi al caso peggiore in situazioni particolari, l'implementazione corretta e l'ordine efficace di esplorazione dei nodi, generalmente conducono a un significativo risparmio di risorse computazionali rispetto al metodo senza potatura alfa-beta.

2.5.2 Esempio di potatura alfa-beta

Supponiamo di disporre di una posizione iniziale P_0 e che la mossa spetti al Bianco (Figura 2.9). Utilizziamo i seguenti parametri: $b = 2$ (fattore di branching) e $d = 3$ (profondità dell'albero). Seguiamo meticolosamente i vari passaggi dell'algoritmo:

1. per prima cosa valutiamo il primo nodo foglia H che restituisce $f(H) = +5$
2. passiamo al suo nodo genitore D e gli settiamo provvisoriamente i valori $\alpha_D = 4$ ($\beta_D = +\infty$)
3. valutiamo il nodo rimanente di D calcolando prima il restante nodo figlio, con $f(I) = +5$. Poiché $f(I) > f(H)$, allora aggiorniamo $\alpha_D = 5$
4. non ci sono più nodi figli per D; propaghiamo quindi il valore di D al nodo genitore B, aggiornando $\beta_B = 5$, $\alpha_B = -\infty$ (al nodo B la mossa è del Nero, il quale deve agire da minimizzatore)
5. esaminiamo ora il nodo E, il quale eredita da B il valore $\beta_E = 5$; in maniera analoga, valutiamo i nodi foglia J e K, supponendo $f(J) = 10$. Propaghiamo al nodo padre il valore $\alpha_E = 10$
6. $\alpha_E > \beta_E$, quindi possiamo interrompere la analisi relativa ai nodi figli di B, dato che il Nero non sceglierà scegliere la mossa (e) che lo porterebbe ad un punteggio inferiore!
7. dato che abbiamo esaurito i nodi figli di B, possiamo propagare il valore contenuto in B alla radice dell'albero e aggiornare i valori $\alpha_A = 5$ e $\beta_A = +\infty$
8. similmente a quanto visto prima, calcoliamo la funzione di valutazione per i nodi foglia dell'altro ramo principale dell'albero; supponiamo $f(L) = 2$ e scriviamo tale valore nel nodo F; nel ramo restante si ha invece $f(M) = -3$, che essendo più piccolo non può passare al nodo successivo
9. si può propagare il valore 2 al nodo C, il quale si ritrova avere $\alpha_C = 5$ e $\beta_C = 2$; è necessario apportare un altro taglio

10. a questo punto l'algoritmo si può concludere, non essendoci altri nodi da esplorare. Il Bianco può quindi scegliere la mossa b che gli fornisce un punteggio pari a 5.

Questo esempio dimostra come la potatura alfa-beta riduca il numero di nodi valutati a soli 5, rispetto agli 8 che sarebbero stati valutati senza potatura. Nonostante ciò, l'algoritmo garantisce comunque al Bianco di scegliere la mossa ottimale nella posizione considerata.

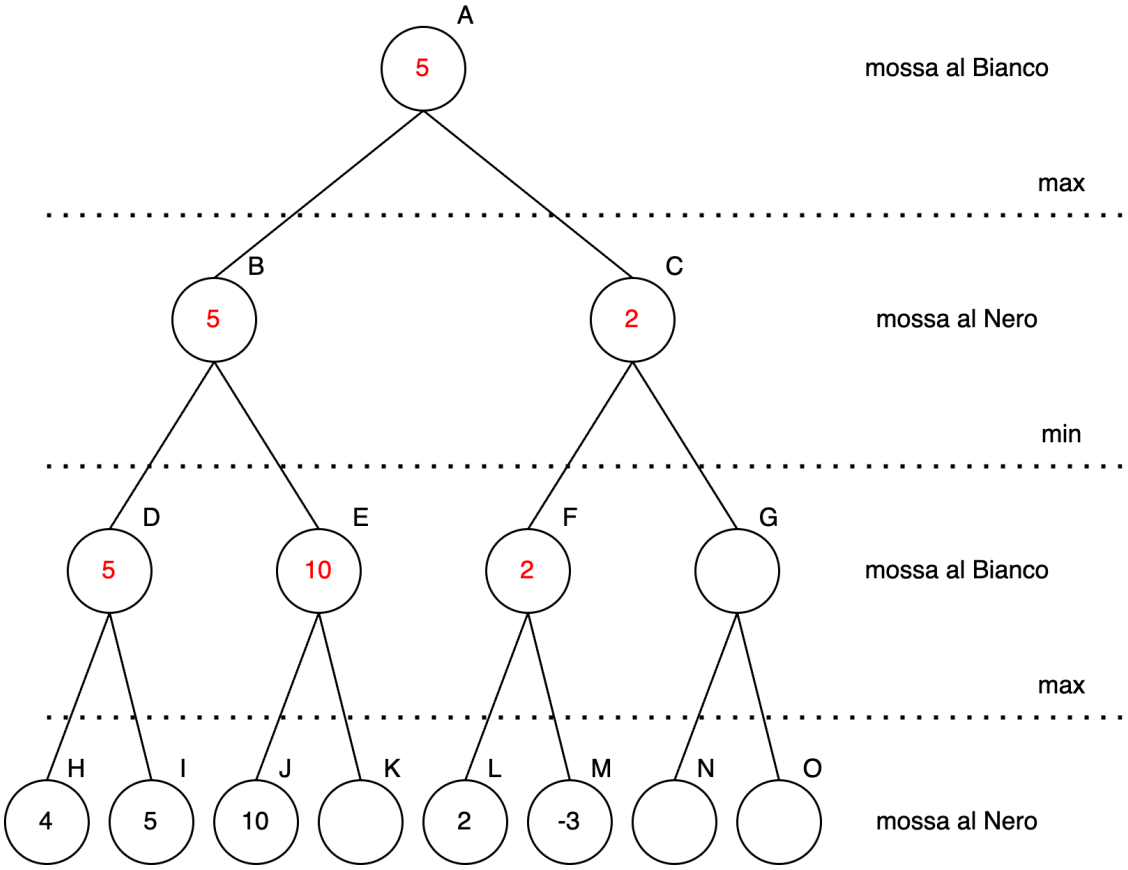


Figura 2.9: Esempio di potatura alfa-beta applicata ad un albero decisionale scacchistico.

2.6 L'avvento dei motori scacchistici in Unione Sovietica

2.6.1 I russi hanno insegnato scacchi al mondo

Nell'Unione Sovietica del secolo scorso, gli scacchi occupavano un ruolo di rilievo nella cultura e nella vita quotidiana. A partire dalla rivoluzione bolscevica, il paese è stato pervaso dalla cosiddetta "febbre scacchistica", che ha coinvolto anche le fasce meno abbienti della popolazione. Lo stato sovietico ha fortemente sostenuto il gioco degli scacchi, considerandolo un eccellente strumento educativo per sviluppare il pensiero critico e strategico. L'URSS ha prodotto una schiera di campioni di fama mondiale, tra cui spiccano nomi illustri come Aleksandr Alekhin, Mikhail Tal, Boris Spassky, Mikhail Botvinnik, fino ai più recenti Anatolij Karpov e Garry Kasparov. Questi grandi maestri hanno dominato la scena internazionale degli scacchi, conquistando numerosi titoli mondiali e consolidando la reputazione dell'URSS come potenza scacchistica globale.

Inizialmente, l'Unione Sovietica guardava con scetticismo all'utilizzo dei computer e delle nuove tecnologie, temendo che potessero rivelarsi una minaccia di minare i diritti dei lavoratori. Tuttavia, questo atteggiamento ha subito un rapido ripensamento che ha accelerato lo studio e la ricerca in questi campi. Già dal 1953, con l'introduzione del primo computer sovietico chiamato *Strela* (in russo *freccia*), il Paese ha inaugurato una nuova era nella ricerca computazionale, inclusa l'informatica e l'analisi dei giochi come gli scacchi. Tuttavia, il vero punto di svolta è arrivato nel 1956 con lo sviluppo del BESM-1, il primo computer pienamente operativo in URSS. Basato su un'architettura a tubi a vuoto, il BESM-1 era in grado di eseguire circa 8.000 operazioni al secondo, rendendolo uno dei computer più potenti e all'avanguardia del suo tempo. Come vedremo, esso ha contribuito significativamente alla competitività dei giocatori sovietici nelle competizioni internazionali di scacchi, supportando campioni come Mikhail Botvinnik e Vasily Smyslov nei loro successi contro avversari di tutto il mondo.

2.6.2 Il primo motore di scacchi russo

Il primo motore di scacchi russo, denominato M-20, è stato sviluppato da un team di ricercatori guidato da Mikhail Shura-Bura (1918-2008) presso l'Istituto di Matematica dell'Accademia delle Scienze dell'URSS. Il nome M-20 deriva dal'omonimo computer, una macchina progettata dall'ingegnere Sergey Lebedev (1959-) e che era uno dei computer più avanzati dell'epoca in Unione Sovietica.

Una breve descrizione del programma è stata fornita in un'intervista con Shura-Bura sugli scacchi al computer, poi pubblicata nell'articolo "La migliore mossa in 58 secondi".[25] Sono inoltre stati pubblicati due giochi del programma.

Il motore di scacchi M-20 era privo di qualsiasi ricerca su albero e includeva serie di funzioni euristiche per la valutazione delle posizioni. Sebbene rudimentale rispetto ai moderni motori di scacchi, l'M-20 era in grado di eseguire analisi e simulazioni di partite, fornendo suggerimenti sulle mosse migliori da effettuare. Tali funzioni includevano sette componenti distinte:

1. **Materiale:** valore assegnato a ogni pezzo (Pedone 1; Cavallo e Alfiere 3.5; Torre 5; Regina 9.5; Re 109).
2. **Mobilità:** numero di mosse legali disponibili per ogni pezzo.
3. **Valutazione dello scambio:** analisi del valore relativo dei pezzi in caso di scambio.
4. **Struttura dei pedoni:** posizionamento e supporto reciproco tra i pedoni dello stesso colore.
5. **Controllo del centro:** presenza e controllo dei pezzi nelle caselle centrali della scacchiera.
6. **Sicurezza del re:** assenza o meno di minacce dirette verso il proprio monarca.
7. **Struttura pedonale davanti il proprio Re:** posizionamento adeguato dei pedoni dell'arrocco.

Il programma di scacchi M-20 utilizzava avanzate tecniche di rappresentazione della scacchiera per migliorare l'efficienza nella generazione e valutazione delle mosse. Un elemento chiave di questa rappresentazione erano le *mappe di attacco e difesa*, conosciute anche come tabelle di attacco. Queste mappe sono strutture dati, spesso implementate come array, che contengono informazioni sull'attacco e la difesa per ogni pedone o pezzo. Inoltre, includono informazioni su quali pezzi controllano, ovvero attaccano o difendono, di ciascuna casella della scacchiera, sia direttamente che indirettamente.

L'uso innovativo di queste tabelle facilitava la valutazione delle posizioni e la scelta delle mosse ottimali. Queste tabelle possono essere considerate come i precursori delle moderne tablebase, ossia di veri e propri database contenenti tutte le mosse possibili derivabili da determinate posizioni nei finali di partita (vedi Paragrafo 3.1).

[25] Herik, J. van den (1983). *Computerschaak, Schaakwereld en Kunstmatige Intelligentie* (Ph.D. thesis). University of Leiden.

2.6.3 La congettura di Botvinnik

Mikhail Botvinnik (1911-1995) è stato un illustre gran maestro[26] sovietico, noto per essere stato il sesto campione del mondo di scacchi, titolo che ha difeso per un record di 15 anni consecutivi, dal 1948 al 1963. Dopo aver perso il titolo contro il connazionale Tigran Petrosian (1929-1984), Botvinnik si è dedicato interamente alla sua scuola di scacchi, rinomata per aver formato altri tre futuri campioni del mondo: Karpov, Kasparov e Kramnik, oltre a numerosi altri grandi maestri di rilievo. Il suo contributo è stato fondamentale nel consolidare il dominio sovietico nel campionato del mondo per più della metà del XX secolo.

Oltre a essere un brillante giocatore di scacchi, Mikhail Botvinnik era anche un ingegnere e manifestava un vivo interesse per l'automazione dei processi di pensiero e le macchine autopensanti. Egli criticava gli algoritmi basati sulla ricerca esaustiva di tutte le possibili varianti dell'albero delle mosse negli scacchi, sostenendo che un giocatore eccellente dovesse concentrarsi su 2 o 3 varianti principali per ogni posizione, utilizzando il proprio intuito e la creatività per scartare rapidamente le alternative meno rilevanti. In questo modo il calcolo si poteva limitare a poche linee e, soprattutto, a poche zone della scacchiera. Secondo Botvinnik, non tutte le 64 caselle avevano la stessa importanza; era sufficiente concentrarsi sulle 10-20 caselle dove potevano verificarsi degli scambi con i pezzi avversari.

Durante la sua vita, Mikhail Botvinnik si dedicò allo sviluppo di un algoritmo completo ed efficiente, concepito per essere tradotto in istruzioni eseguibili da una macchina. Egli riteneva fondamentale che una volta stabilite queste regole di base, fosse necessario creare un programma capace di imparare autonomamente dalle proprie partite giocate. Questo campo, tuttavia, era ancora in fase embrionale nei suoi anni di ricerca e sviluppo.

In virtù della sua formazione di ingegnere, Mikhail Botvinnik ha sviluppato le innovative *tabelle di codifica*, una particolare rappresentazione matematica che si concentrava sul movimento dei pezzi all'interno di una posizione, fornendo un quadro sistemico per comprendere e analizzare le dinamiche del gioco. Queste tabelle permettevano di codificare e memorizzare in modo efficiente le informazioni relative a una determinata disposizione dei pezzi, facilitando lo studio e l'analisi strategica delle partite da parte dei computer.

Botvinnik era così fiducioso nelle potenzialità degli sviluppi tecnologici che ha predetto che prima della sua morte avrebbe assistito sicuramente ad un campionato

[26]Il titolo di Grande Maestro Internazionale, abbreviato spesso in GM, rappresenta il più alto riconoscimento che la Federazione Internazionale degli Scacchi (FIDE) possa conferire a un giocatore di scacchi. Questo titolo è a vita ed è riconosciuto a livello globale.

del mondo di scacchi giocato tra computer.

2.6.4 La guerra fredda . . . sulla scacchiera

Siamo giunti cronologicamente al 22 novembre 1966, una data fondamentale perché segna il primo storico scontro tra computer nel mondo degli scacchi! Questa sfida prevedeva il gioco contemporaneo di quattro partite per corrispondenza,[27] estese su nove mesi, tra il *Kotok-McCarthy-Program* degli Stati Uniti, operante su un computer IBM 7090, e l'*ITEP Chess Program* dell'Unione Sovietica, che utilizzava un M-20. Questo confronto inizialmente proposto da John McCarthy durante una visita in Unione Sovietica nel 1965, rappresentava una sfida scientifica e tecnologica durante la tesissima Guerra Fredda tra USA e URSS. Entrambi i programmi erano supportati da un hardware potente per l'epoca, con prestazioni di calcolo confrontabili intorno alle 20.000 istruzioni al secondo. Queste macchine occupavano interamente una sala spaziosa, simbolo delle risorse e dell'impegno investiti da entrambe le nazioni nella corsa per la supremazia mondiale.

Il programma sovietico, sviluppato da un gruppo di eccelsi matematici guidati da Georgy Adelson-Velsky e con la consulenza del maestro[28] russo Alexander Bitman, ha dimostrato una sorprendente superiorità, nonostante fosse eseguito su hardware leggermente meno avanzato. La vittoria dell'*ITEP Chess Program* con un punteggio di 3-1 ha evidenziato la sua capacità di sfruttare le vulnerabilità tattiche del *Kotok-McCarthy-Program*, derivanti probabilmente da una non ottimale implementazione della strategia di Shannon di tipo B, in contrasto con la più sofisticata e precisa struttura basata sulla strategia di Shannon di tipo A adottata dal programma sovietico (vedi Paragrafi 2.4.3 e 2.4.4).

Questa sconfitta, occorsa in un periodo di intensa competizione tecnologica e scientifica, fu percepita come un'umiliazione nazionale negli Stati Uniti, paragonabile alla corsa per la conquista della Luna. Alla fine dell'incontro, il *New York Times* ha recitato «Eletronic Chess is Won by Soviet». L'esito negativo ha stimolato ulteriori ricerche e innovazioni nel campo dell'informatica e dell'intelligenza artificiale in tutto il Paese, spingendo gli Stati Uniti a intensificare gli sforzi per recuperare il divario tecnologico.

[27] Una partita a scacchi per corrispondenza è una partita dove i giocatori possono muovere i pezzi senza la necessità di essere presenti fisicamente allo stesso tavolo. Questo tipo di partite permette ai giocatori di pianificare le loro mosse con calma, di solito con lunghi periodi di riflessione tra una mossa e l'altra. Le mosse possono essere scambiate attraverso vari mezzi: posta elettronica, posta normale o piattaforme online.

[28] Il titolo di Maestro FIDE (FM) può essere ottenuto da un giocatore che raggiunge i 2300 punti Elo FIDE a cadenza classica (o standard) nelle competizioni internazionali.

2.7 La nuova generazione

Analizzeremo brevemente in questo paragrafo l'evoluzione dei motori scacchistici a partire dalla fine degli anni '60 (a partire dalla sfida tra USA e URSS) fino ai primi anni 2000, sia in America che in Unione Sovietica.

2.7.1 MacHack

Dopo l'umiliazione subita dai rivali orientali, gli americani non hanno esitato a includere nel loro team di ricerca il genio della programmazione, nonché il fondatore della comunità hacker, Richard Greenblatt (1944-). Greenblatt ha individuato rapidamente le difficoltà del programma di Kotok e McCarthy, attribuendole a un errore concettuale: il programma utilizzava la strategia di Shannon di tipo B ma con un numero limitato di mosse alternative. In particolare, considerava 4 mosse al primo livello di ricerca, 3 al secondo, 2 al terzo e quarto, e solo una al quinto. Probabilmente, la loro mancanza di esperienza nel gioco degli scacchi li portò a non capire che ridurre drasticamente la profondità delle mosse era dannoso per il computer, poiché gli impediva di riconoscere i colpi tattici dell'avversario.

Greenblatt è riuscito a migliorare il programma aumentando il numero di alternative valutate in una determinata posizione e ampliando i livelli di ricerca del motore. Questo nuovo programma, conosciuto come *Mac Hack*, è stato il primo computer ad essere introdotto in un torneo FIDE ufficiale per umani, dando del filo da torcere a molti appassionati del settore.

2.7.2 Kaissa

In Unione Sovietica, nonostante la schiacciante vittoria contro i rivali storici, i ricercatori erano divisi in due correnti di pensiero differenti: quella accademica e quella guidata da Mikhail Botvinnik.

Il gruppo accademico riuscì a sviluppare nel 1972 il programma *Kaissa*, il cui nome deriva da una ninfa mitologica greca, considerata la dea degli scacchi. Kaissa utilizzava tecniche avanzate per l'epoca, tra cui l'algoritmo minimax con potatura alpha-beta e il *metodo delle analogie*. Quest'ultimo era un metodo di potatura che attribuiva la stessa valutazione a posizioni simili, riducendo significativamente il costo computazionale della ricerca.

Questo programma si distinse rapidamente nel panorama degli scacchi computerizzati, vincendo il primo Campionato del Mondo di Scacchi per Computer che si tenne a Stoccolma nel 1974.

2.7.3 Pioneer

Parallelamente allo sviluppo dei software da parte degli accademici di Mosca, Botvinnik si dedicava alla traduzione in codice del suo ambizioso progetto di realizzare un programma di scacchi capace di giocare come un gran maestro. Questo progetto richiedeva la trasposizione delle sue sofisticate idee sul calcolo delle traiettorie e sulle *zone* di attività sulla scacchiera in codice informatico. In particolare, il concetto di zone si riferisce a parti della scacchiera caratterizzate da un'intensa attività dei pezzi, che implicano potenziali scambi e minacce da considerare. Nonostante lo scetticismo e la contrarietà di molti ricercatori dell'epoca, Botvinnik trovò un alleato nel giovane Vladimir Butenko. Nel 1967, Butenko implementò per la prima volta la rappresentazione della scheda degli attacchi vettoriali 15×15 su un computer M-20. Questa rappresentazione consentiva di determinare le traiettorie dei pezzi sulla scacchiera, sovrapponendo una matrice 15×15 a una scacchiera 8×8, in modo che il quadrato centrale della matrice coincidesse con una casella chiave, fungendo da origine per tutti i vettori di spostamento.

Nella proiezione matematica gerarchica (MP) di Botvinnik, gli scacchi erano visti come un sistema complesso articolato su diversi livelli. Al livello più basso vi erano le traiettorie, rappresentanti i percorsi possibili per i pezzi. Il livello intermedio comprendeva le zone, una rete di traiettorie principali legate a piani di attacco o difesa. Queste traiettorie principali potevano essere contrastate da contro-traiettorie dell'avversario e supportate da contro-traiettorie di supporto.

Negli anni '70 e '80, un nuovo team di ricerca collaborò con Botvinnik sul progetto *Pioneer* presso l'Istituto Nazionale di Ricerca per l'Ingegneria Elettrica e l'Accademia delle Scienze dell'URSS a Mosca. Il team sviluppò una strategia di generazione di mosse basata su mappe di attacco e di offset, utilizzando attacchi vettoriali e un modello geometrico gerarchico. Questo modello era composto da traiettorie, fasci di traiettorie e catene di pezzi, ciascuno con le proprie traiettorie, che cooperavano per supportare o ostacolare gli attacchi contro un particolare bersaglio.

In conclusione, il programma Pioneer riusciva a risolvere problemi di scacchi, inclusi quelli più complessi. Tuttavia, non era ancora competitivo nei tornei di scacchi contro altri computer, poiché commetteva numerosi errori tattici in diverse situazioni di gioco.

2.7.4 Belle

Tra i tanti geni dell'informatica che si sono addentrati nello sviluppo delle tecnologie per gli scacchi, non poteva mancare Ken Thompson (1943-), uno dei precursori della

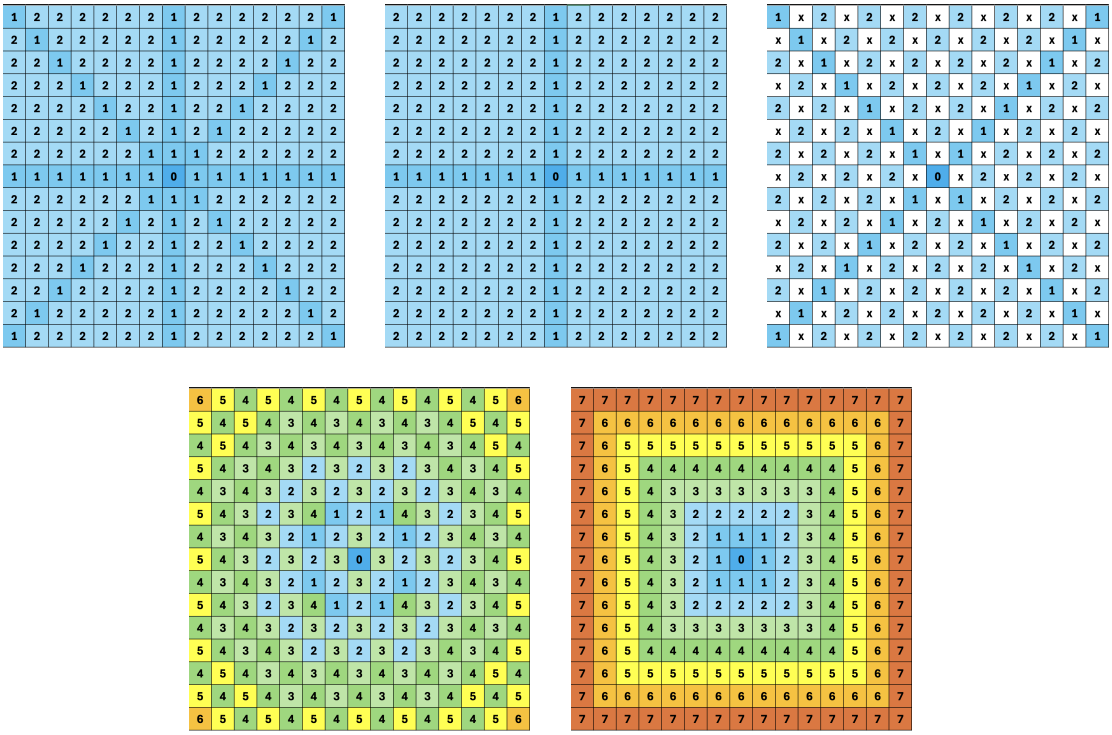


Figura 2.10: Esempi di tabelle di codifica 15×15 colorate per donna, torre, alfiere, cavallo e re. La velocità del movimento del pezzo diminuisce all'aumentare dell'intensità del colore rosso. L'alfiere presenta caselle bianche (segnate con una x) in quanto può occupare solo metà scacchiera.

programmazione e padre del sistema operativo UNIX. Nel 1977, Thompson creò *belle*, uno dei primi motori di scacchi con un hardware appositamente progettato per questo scopo. Questo programma innovativo era in grado di giocare a un livello competitivo per l'epoca, diventando il più forte computer scacchistico del suo tempo. *Belle* raggiunse nel 1983 un Elo[29] di 2250, diventando la prima macchina in grado di giocare al livello di un maestro. Ha vinto il Campionato del Mondo di Scacchi per Computer (WCCC) nel 1980 e il North American Computer Chess Championship in ben cinque occasioni.

Nella sua configurazione finale, *belle* era costituito da una macchina PDP-11/23, un processore LSI-11 e diverse schede custom. Tre schede erano dedicate alla generazione delle mosse, quattro alla valutazione delle posizioni, e una fungeva da controller per l'intero sistema. Il sistema disponeva di un megabyte di memoria commerciale, utilizzata per le *transposition table*.[30]

2.7.5 Deep Blue

Le origini di Deep Blue risalgono a metà degli anni '80, quando due studenti di dottorato alla Carnegie Mellon University, Feng-hsiung Hsu e Murray Campbell, svilupparono un prototipo di motore di scacchi chiamato *ChipTest*. Questo progetto mirava a creare un chip hardware dedicato specificamente alla valutazione delle posizioni scacchistiche e si voleva rifare al progetto Belle di Ken Thompson.

ChipTest evolse in un progetto più avanzato chiamato Deep Thought. Nel 1989, Deep Thought vinse il WCCC. Impressionata dai successi di Deep Thought, IBM reclutò Hsu e Campbell, fornendo loro le risorse necessarie per sviluppare un sistema di scacchi ancora più potente, che divenne noto come *Deep Blue*.

Deep Blue era un sistema di calcolo parallelo costituito da un supercomputer IBM RS/6000 dotato di 480 processori personalizzati per l'elaborazione delle mosse scacchistiche. Ogni processore poteva valutare milioni di posizioni al secondo, consentendo a Deep Blue di analizzare una vasta gamma di mosse e contromosse in tempi molto ridotti, fino a un massimo di 200 milioni di mosse al secondo. L'algo-

[29]Il punteggio Elo fu ideato dall'ungherese Arpad Elo (1903-1992) ed è un sistema numerico utilizzato per valutare il livello di abilità di un giocatore di scacchi rispetto ad altri giocatori. Più è alto il punteggio Elo, più lo scacchista è forte. Ad ogni partita disputata, il punteggio viene aggiornato in base al risultato e al punteggio dell'avversario.

[30]Una transposition table è una struttura dati utilizzata nei motori di scacchi e in altri giochi a somma zero per memorizzare le posizioni già valutate durante la ricerca. È tipicamente implementata come una hash table.

ritmo di gioco degli scacchi era scritto in linguaggio C e funzionava su un sistema operativo AIX.[31]

Questo motore divenne noto principalmente per essere stato progettato con l'obiettivo dichiarato di sfidare e sconfiggere il campione del mondo umano di scacchi in carica. Effettivamente, non molto dopo, la sfida stava per avere inizio!

2.8 Lo scontro che ha sconvolto la storia

2.8.1 Kasparov: il difensore della razza umana

Nel 1996, si è svolta la prima sfida storica tra Deep Blue e il campione del mondo Garry Kasparov. Prima dell'incontro, Kasparov ha dichiarato con convinzione: «Difenderò la razza umana!». Quella sfida non è stata soltanto un semplice confronto a tavolino, ma si è trasformata rapidamente in una lotta epica per la sopravvivenza, dove la superiorità delle macchine non avrebbe dovuto prevalere sul genio del loro creatore. Era il momento in cui l'umanità doveva riaffermare la propria supremazia intellettuale nei confronti di un pretendente sempre più potente e impetuoso.

Kasparov si è impegnato con tutta la sua abilità e tenacia per difendere il primato umano (e anche il suo), facendo appello a ogni risorsa strategica e tattica a sua disposizione. Alla fine delle sei partite disputate, Kasparov ha trionfato nel match con un punteggio complessivo di 4-2, ma clamorosamente perdendo una partita.[32] La razza umana era al sicuro, ma quella sconfitta ha destato anche i più scettici ... I tempi erano ormai maturi perché l'intelligenza scacchistica artificiale potesse superare l'uomo? Era questo l'inizio di una nuova era, o è stato solo un colpo di fortuna?

2.8.2 Il ritorno dell'IA: la rivincita di Deep Blue

Dopo la prima sfida persa, il team di Deep Blue non si è arreso e ha continuato a lavorare intensamente per aggiornare e perfezionare il motore scacchistico, modellando il sistema allo stile di gioco di Kasparov e correggendo alcuni errori commessi nel precedente incontro.

Nel maggio 1997, una versione migliorata di Deep Blue (non ufficialmente soprannominata *Deeper Blue*) ha avuto l'opportunità di una rivincita contro il suo acer-

[31] AIX (*Advanced Interactive eXecutive*) è un sistema operativo UNIX sviluppato da IBM, progettato per offrire affidabilità, sicurezza e scalabilità avanzate per server e workstation IBM Power Systems.

[32] Al pareggio viene solitamente attribuito un punteggio di 0.5 per ciascuno dei due giocatori.

rimo avversario Garry Kasparov, affrontandolo nuovamente in un secondo incontro storico.

Figura 2.11: Garry Kasparov impegnato nella sfida contro Deep Blue nel 1997. Autore: Adam Nadel. Fonte: AP1997 ©

La sfida sembrava iniziare splendidamente per il campione russo, che ha vinto la partita d'esordio con i pezzi bianchi. Pareva infatti che avesse preso le misure e avesse capito come affrontare efficacemente un'intelligenza artificiale, basata su un algoritmo di calcolo bruto e di ricerca su un vastissimo albero decisionale. Kasparov aveva immediatamente intuito che doveva privilegiare posizioni chiuse e con pochi scambi di pezzi (più strategiche) anziché posizioni aperte che offrono la possibilità di cambi e minacce (più tattiche).

Questo approccio ha obbligato Kasparov ad adattarsi, abbandonando in parte il suo stile di gioco tattico e aggressivo. Ha dovuto concentrarsi su posizioni più tranquille e pianificare strategie a medio e lungo termine, aspetti che i motori di scacchi di quel tempo non riuscivano a gestire perfettamente. Per questo motivo, ha dovuto più volte astenersi a giocare la sua apertura preferita, la Difesa Siciliana, nota per la sua complessità e dinamicità, poiché questa avrebbe favorito lo stile di gioco del computer. Invece, si è limitato infatti ad aperture più posizionali quali la Difesa Spagnola o la Difesa Caro-Kann.

La seconda partita del match può essere considerata come un vero e proprio punto di svolta: sebbene Kasparov sembrasse sicuro dei suoi mezzi, non poteva ancora prevedere ciò che lo avrebbe atteso... Vediamo di seguito il momento saliente:

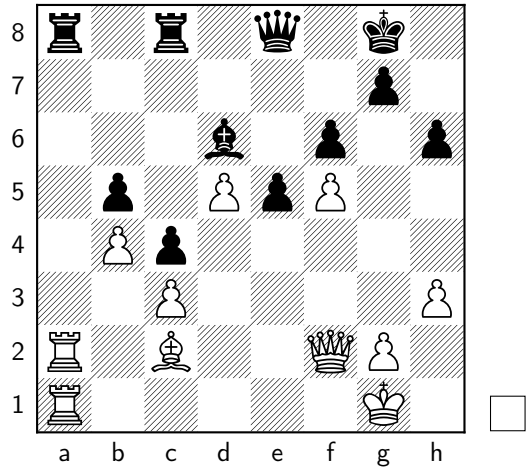


Figura 2.12: IBM Deep Blue - G. Kasparov, posizione sulla scacchiera alla mossa 36, 2° turno, 1997.

C'è da precisare che sia Deep Blue (con i Bianchi) che Kasparov (con i Neri) non hanno sempre giocato in modo impeccabile e hanno commesso entrambi alcune imprecisioni. Ritornando alla partita, osserviamo la posizione alla mossa 36, dove il turno è quello del Bianco. Kasparov si aspettava dal suo avversario la tattica e logica **37 ♕b6**, con attacco doppio all'alfiere in d6 e al pedone in b5, una manovra tattica che un calcolatore avrebbe potuto vedere senza troppe difficoltà. Kasparov non avrebbe avuto problemi a fronteggiare questa minaccia: avrebbe infatti volontariamente ceduto il proprio alfiere in cambio dell'attivazione immediata della propria donna e di un'inevitabile scacco perpetuo,[33] situazione che avrebbe condotto la partita ad una patta assicurata.

Tuttavia, con grande stupore di Kasparov, Deep Blue ha giocato la brillante **37 ♗e4!**, mossa che non mira ad alcun vantaggio tattico immediato, ma piuttosto a un obiettivo strategico a lungo termine: l'alfiere funge da bloccatore per il pedone nero in e5, che è passato, e potrebbe in qualsiasi momento avanzare e creare scompigli nelle retroguardie del Bianco. Probabilmente, questa mossa potrebbe aver condotto Kasparov ad un vero e proprio blocco psicologico, portandolo ad autoconvincersi che il computer avesse ricevuto un aiutino da qualche essere umano. Il fatto che Deep Blue fosse situato a diversi chilometri dalla scacchiera ha inoltre alimentato questi sospetti.

Purtroppo, Kasparov non ha tenuto conto di un grande svantaggio che lui ha rispetto al computer: le emozioni. Queste possono influenzare significativamente il

[33]Lo scacco perpetuo è un caso particolare di patta per triplice ripetizione, in cui un giocatore forza la ripetizione tramite una serie di scacchi. Questo è spesso utilizzato come strategia difensiva dal giocatore in svantaggio, consentendogli di pareggiare una partita che altrimenti sarebbe persa.

pensiero razionale e le decisioni durante una partita di scacchi. La mente umana può giocare brutti scherzi, perdendo lucidità nei momenti più critici. Dopo aver permesso a Deep Blue di pareggiare il punteggio del match, Kasparov non è riuscito più a risollevarsi, e ha subito il peso psicologico del confronto.

Il match è proseguito con 3 pareggi consecutivi, giungendo alla sesta e ultima partita in perfetta parità. Kasparov, giocando con i pezzi neri, si trovava con lo svantaggio del tratto[34] e con la pressione di non deludere l'aspettativa di milioni di persone.

La partita è iniziata male per il campione russo che, dopo solo 7 mosse, ha commesso un'insolita svista che ha permesso a Deep Blue di sacrificare un cavallo, danneggiando la struttura pedonale del Nero e aprendo numerose linee di attacco verso il suo re.

La posizione si è presto compromessa: i pezzi neri di Kasparov arretravano uno dopo l'altro, mentre quelli bianchi di Deep Blue si attivavano sempre più rapidamente. Alla mossa 18, Kasparov si è trovato costretto a sacrificare la propria regina nel tentativo di alleggerire la posizione, ma questo non è stato sufficiente. Dopo poche mosse, ha deciso di gettare la spugna. La tensione mentale aveva avuto la meglio sul più grande scacchista del mondo.

La sconfitta di Kasparov ha suscitato un enorme scalpore, anche al di fuori degli ambienti scacchistici: per la prima volta, un computer è riuscito a battere il campione del mondo! L'era degli uomini è giunta al termine, lasciando il posto a quella delle macchine ...

[34]Il Bianco, muovendo per primo, inizia la partita con un leggero vantaggio, conosciuto come "vantaggio del tratto".

Capitolo 3

L'avvento dell'Intelligenza Artificiale e delle reti neurali

Dopo la storica vittoria di IBM Deep Blue contro Garry Kasparov nel 1997, l'evoluzione dei motori di scacchi ha subito una rapida accelerazione, portando alla nascita di numerosi programmi sempre più potenti e sofisticati. Deep Blue ha rappresentato una pietra miliare e ha inaugurato un'era in cui i motori di scacchi avrebbero dominato incontrastati l'intero panorama mondiale.

3.1 La nuova era degli scacchi: l'egemonia dei motori moderni

L'aumento della potenza hardware, insieme all'uso del calcolo parallelo e delle GPU, ha permesso ai motori moderni di analizzare milioni di posizioni al secondo. Inoltre, l'integrazione di vasti database di aperture e tablebase di finali ha consentono alle macchine di giocare in modo ottimale in molte situazioni pre-calcolate.

- Le prime mosse d'apertura sono praticamente già note ai moderni motori di scacchi, i quali non hanno la necessità di calcolare varianti complesse sin dalle prime mosse. Invece, essi si trovano a dover scegliere la mossa più adatta tra una serie di opzioni già preimpostate. Quando più mosse hanno un valore equivalente, la scelta può essere effettuata mediante algoritmi casuali per garantire varietà nelle partite.

- I finali sono una sfida ardua anche per i giocatori più esperti, poiché richiedono manovre lunghe e precise. In questa fase della partita, è fondamentale riconoscere tutte le possibili posizioni che possono condurre a un vantaggio

concreto. Per questo motivo sono state ideate le *tablebase*, particolari database che memorizzano ogni posizione possibile di un finale specifico e la mossa migliore per forzare la vittoria. Le tablebase utilizzano le bitboard, array di 64 bit che rappresentano facilmente una posizione di scacchi per un motore, utili per gestire e accedere rapidamente a queste informazioni. Le soluzioni di questi problemi vengono determinate a ritroso a partire dalle posizioni finali (di scacco matto o di stallo),[1] trovando progressivamente la sequenza di mosse ottimali.

numero di pezzi	numero di posizioni
2	462
3	368.079
4	125.246.598
5	25.912.594.054
6	43.787.154.440.416
7	423.836.835.667.331
8	38.176.306.877.748.245
...	...

Tabella 3.1: Numero di posizioni possibili da analizzare in un finale di scacchi con un determinato numero di pezzi.

3.1.1 Fritz

Fritz è una serie di programmi di scacchi pubblicati dall'azienda ChessBase, originariamente sviluppati da Frans Morsch e Mathias Feist. Da menzionare particolarmente è Fritz 18 Neuronal, rilasciato nel marzo 2022 come aggiornamento gratuito con l'introduzione della valutazione NNUE, una caratteristica ora diffusa tra i motori di scacchi moderni (vedi Paragrafo 3.3).
Una delle innovazioni più significative di Fritz è l'implementazione della "ricerca a movimento nullo" (*null-move pruning*), una tecnica avanzata mirata a migliorare l'efficienza della ricerca delle mosse. Questa tecnica implica la simulazione di una "mossa nulla", permettendo all'avversario di muovere due volte consecutivamente. Il motore valuta quindi se la posizione risultante è vantaggiosa per l'avversario: se lo è in modo significativo, il motore scarta la posizione originale come debole senza ulteriori analisi, risparmiando tempo di calcolo. Se la posizione non mostra

[1] Lo stallo negli scacchi è una situazione in cui il giocatore a cui spetta il turno di gioco non ha mosse legali disponibili e non ha il proprio re sotto scacco. In questo caso la partita termina immediatamente in parità, indipendentemente dal materiale rimanente sulla scacchiera. Lo stallo è considerato una strategia difensiva utile per il giocatore in svantaggio, poiché può trasformare una sconfitta imminente in un pareggio.

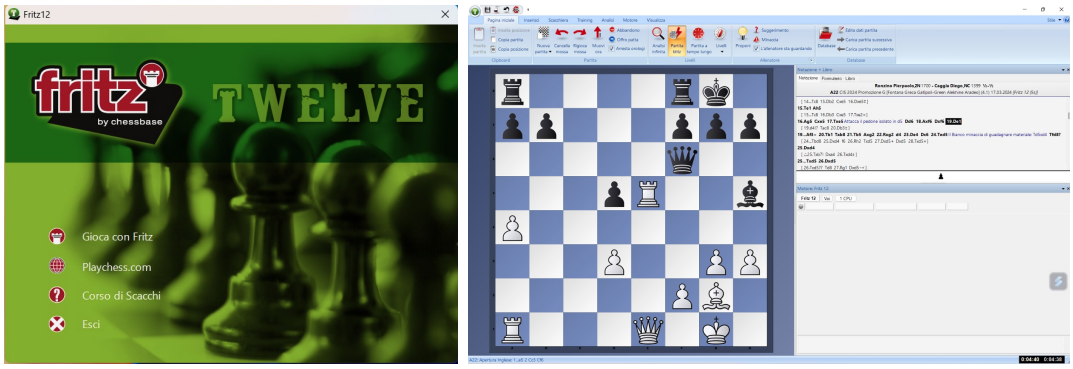


Figura 3.1: Interfaccia grafica del motore Fritz 12.

un vantaggio rilevante per l'avversario, il motore continua l'analisi in modo più approfondito.

3.1.2 Crafty

Crafty è un motore di scacchi open source scritto da Robert Hyatt negli anni '90. La novità principale è l'utilizzo delle *rotated bitboard* e successivamente delle *magic bitboard*, struttre di dati che semplificano il codice senza comprometterne la velocità. Il motore implementa una ricerca parallela avanzata (Generation II) che migliora la velocità e l'efficienza dell'analisi e impiega tecniche avanzate come il null move pruning e il *Late Move Reduction* (LMR) per ridurre la ricerca delle mosse meno promettenti.

3.1.3 Shredder

Shredder, sviluppato da Stefan Meyer-Kahlen nel 1993, è un motore di scacchi che ha vinto numerosi tornei e campionati mondiali di scacchi per computer. Ha debuttato al WMCCC 1995 a Paderborn, vincendo il suo primo titolo un anno dopo. Shredder è stato distribuito commercialmente da Millennium 2000 GmbH e, dalla versione 5, supporta il protocollo UCI, permettendogli di usare qualsiasi interfaccia grafica compatibile. Shredder è scritto in ANSI-C, rendendolo facilmente compilabile su varie piattaforme hardware. La versione Shredder 13 e la sua controparte multiprocessore Deep Shredder 13 sono state rilasciate il 30 ottobre 2016. Grazie ai suoi eccellenti risultati nei tornei e alle sue capacità di adattamento, Shredder è considerato uno dei programmi di scacchi per microcomputer più forti in circolazione.

3.1.4 Stockfish

L'ultimo motore (ma non per importanza) che illustriamo brevemente in questo paragrafo è Stockfish, un motore scacchistico open source compatibile con il protocollo UCI, sviluppato da Tord Romstad, Marco Costalba, Joona Kiiski e Gary Linscott. È scritto in C++, può essere compilato per vari processori e sistemi operativi.

Inizialmente tra i primi venti motori scacchistici, Stockfish ha rapidamente scalato le classifiche, diventando il motore più forte disponibile al pubblico nel 2018.

Nel 2011, Marco Costalba e Joona Kiiski si sono ritirati come manutentori del progetto, lasciando lo sviluppo e la manutenzione alla comunità open source di Stockfish. Una sinergia con la comunità del Shogi ha portato alla creazione del ramo Stockfish NNUE, che ha introdotto le reti neurali nel 2019. Stockfish 14, rilasciato il 2 luglio 2021, ha visto ulteriori miglioramenti dovuti alla progettazione di una nuova architettura NNUE insieme a un trainer accelerato dalla GPU scritto in PyTorch. Inoltre, gli sono state fornite miliardi di posizioni per addestrare il nuovo NNUE. Stockfish 16, rilasciato il 30 giugno 2023, ha eliminato la valutazione classica, concentrandosi esclusivamente sulle reti neurali NNUE.

Figura 3.2: Logo di Stockfish. Fonte: stockfishchess.org.

Stockfish è un contendente di primo piano nel *Top Chess Engines Competition* (TCEC),[2] raggiungendo le superfinali dalla stagione 4 e stabilendo il suo status di numero uno mondiale vincendo diversi titoli, lasciando indietro rivali commerciali come Komodo e Houdini.

Nel giugno del 2023, Stockfish domina la lista CCRL[3] con un rating di 3530 punti Elo, il più alto tra i motori di scacchi al mondo. Per mettere questo punteggio in prospettiva, basti considerare che l'attuale numero uno al mondo degli scacchi umani, Magnus Carlsen, ha un rating Elo di 2832 (al momento della stesura di

[2]Il TCEC è un prestigioso torneo per motori scacchistici fondato nel 2010. Si svolge in più stagioni annue, con fasi preliminari, fasi a gironi, una fase finale e una Superfinale tra i due migliori motori. Tutti i motori competono su hardware identico per garantire equità.

[3]La lista CCRL (*Computer Chess Rating Lists*) è una classifica che valuta e confronta la forza dei motori di scacchi in base ad un ampio database di partite giocate tra di loro.

questa tesi).[4] Questo confronto evidenzia come i motori di scacchi abbiano preso il sopravvento nel panorama internazionale rispetto agli esseri umani.

3.2 La rivoluzione di AlphaZero

In questo paragrafo assisteremo ad una vera e propria rivoluzione nel gioco degli scacchi, che ha messo in discussione sia secoli di tradizione e dogmi dettati dai grandi campioni del passato, sia decenni di sviluppo di programmi scacchistici sempre più avanzati e complessi.

3.2.1 Il gioco di un extraterrestre

Il 5 dicembre 2017, la società londinese di intelligenza artificiale DeepMind ha pubblicato un documento intitolato "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". Questo studio ha descritto le straordinarie capacità di AlphaZero, un'intelligenza artificiale auto-apprendente sviluppata da DeepMind. AlphaZero, partendo unicamente dalle regole dei giochi e giocando partite casuali contro sé stesso, ha raggiunto in sole 24 ore un livello sovrumano negli scacchi, nello shogi (scacchi giapponesi) e nel Go (un altro gioco da tavolo inventato in Cina). Ha ottenuto vittorie notevoli contro i migliori programmi esistenti per ciascun gioco, annoverando tra queste una schiacciante vittoria contro Stockfish, il campione mondiale dei motori scacchistici (vedi Paragrafo 3.4).

Questo rappresenta un traguardo senza precedenti, poiché AlphaZero è il primo programma di scacchi a raggiungere una forza sovrumana interamente attraverso l'auto-apprendimento, senza impiegare alcun database di partite umane, e senza alcuna formula di valutazione ideata dall'uomo. In un certo senso, è come se stessimo osservando il gioco di un'entità extraterrestre giunta sulla Terra!

In un'intervista successiva contenuta in questo lavoro, Demis Hassabis, cofondatore di DeepMind, discute di come il successo di AlphaZero sia stato costruito sui precedenti traguardi raggiunti con AlphaGo. Quest'ultimo, utilizzando reti neurali e *deep learning*, aveva sconfitto il campione di Go Lee Sedol nel 2016. Entrambi questi traguardi segnano importanti tappe nel percorso di DeepMind verso l'utilizzo dell'intelligenza artificiale a vantaggio dell'umanità.

[4]Per visionare la classifica aggiornata in tempo reale dei giocatori con il miglior punteggio Elo, visitare il sito www.chess.com/ratings.

3.2.2 Caratteristiche principali del motore

Come abbiamo visto nel Capitolo 2, i motori tradizionali utilizzano un albero di ricerca con potatura alfa-beta combinato con una funzione di valutazione progettata da esperti giocatori di scacchi. AlphaZero, al contrario, impara completamente da solo, sviluppando autonomamente la propria funzione di valutazione e utilizzando un tipo differente di ricerca, di tipo probabilistica, chiamata *ricerca Monte Carlo* (MCTS). Questa tecnica si avvale della conoscenza acquisita in precedenza per identificare le mosse che richiedono maggiore attenzione. L'output delle reti neurali orienta la ricerca Monte Carlo verso i rami più promettenti dell'albero di decisione.

Riguardo la funzione di valutazione, se ne occupa la rete neurale di AlphaZero, un sistema informatico che ricorda le connessioni dei neuroni del cervello (vedi Paragrafo 3.3). La rete neurale riceve in input la posizione attuale della partita e restituisce:

1. una lista di mosse suggerite, accostate alla probabilità di essere effettivamente eseguite,

2. una stima percentuale dell'esito atteso per la posizione corrente.

Nei motori moderni di scacchi, il punteggio di valutazione solitamente indica un vantaggio in termini di pedoni o equivalenti pedoni che uno dei giocatori ha rispetto all'altro. Ad esempio, un punteggio di $+1.00$ indicherebbe un vantaggio di un pedone per il Bianco, come un punteggio di -6.50 denoterebbe un vantaggio di circa sei pedoni e mezzo per il Nero. D'altra parte, AlphaZero utilizza un approccio diverso. Non fornisce un punteggio diretto in termini di pedoni, ma stima la probabilità di vittoria per un giocatore in una data posizione. Questo valore è espresso in una scala da 0 a 1, dove 1 rappresenta la certezza della vittoria, 0.50 indica una perfetta parità e 0 indica la certezza della sconfitta. Questa metodologia permette ad AlphaZero di valutare le posizioni in modo più probabilistico e di focalizzarsi sulle strategie che massimizzano le sue possibilità di vincita anziché su un vantaggio materiale diretto come i pedoni.

È interessante notare che lo stesso algoritmo è in grado di raggiungere capacità sovrumane anche in altri giochi, senza la necessità di adattare l'architettura per ciascun contesto specifico. In altre parole, tale sistema mostra un certo grado di generalità: lo stesso processo di ricerca nell'albero Monte Carlo guidato da una rete neurale, addestrato con apprendimento per rinforzo tramite *self-play*, si dimostra efficace in diverse applicazioni senza necessità di impostazioni o modifiche specifiche del gioco in questione.

3.2.3 Il training di AlphaZero

I grandi maestri di scacchi, in preparazione per un match imminente, dedicano molte ore, talvolta giorni, per la preparazione di un certo repertorio di aperture e per lo studio di vari sistemi di gioco da attuare. In particolare, si preparano attentamente per il loro imminente avversario, analizzando tutte le sue partite già giocate e registrate nel vasto database dei tornei mondiali, cercando accuratamente le debolezze nel suo stile di gioco e studiando con dettaglio ogni piano per arginare i suoi punti di forza.

Al contrario, l'addestramento di AlphaZero prima del match con Stockfish è durato solo nove ore! Il motore è partito da zero, senza alcuna conoscenza pregressa degli scacchi oltre alle regole di base, e non ha studiato il gioco del suo avversario. Durante l'addestramento, AlphaZero non ha utilizzato alcun database di aperture, sviluppando il proprio repertorio esclusivamente attraverso partite contro sé stesso.

All'inizio di queste nove ore, AlphaZero ha iniziato a giocare effettuando mosse casuali, terminando quasi tutte le partite in parità. Tuttavia, con il tempo, alcune partite hanno iniziato a concludersi con una vittoria per una delle due parti. Questo punto di svolta ha permesso ad AlphaZero di migliorare la propria funzione di valutazione, aggiornando continuamente i pesi e altri parametri della rete neurale.

Durante questo processo di apprendimento, il sistema regolava costantemente i parametri della rete neurale per rendere più probabili le mosse che portavano alla vittoria, valutando le posizioni favorevoli per il giocatore vincente. Questo continuo aggiustamento ha reso AlphaZero progressivamente più potente, fino a diventare un avversario imbattibile.

Sempre durante queste fatidiche nove ore, AlphaZero ha affrontato un totale di 44 milioni di partite contro sé stesso, operando a una velocità di circa 1.000 partite al secondo. Durante questo processo, ha costantemente adattato i parametri della sua rete neurale per riflettere le mosse e gli esiti delle partite più recenti. Per ogni mossa giocata durante il self-play, il MCTS ha eseguito 800 simulazioni, ciascuna estendendo la ricerca corrente per valutare il valore delle posizioni risultanti.

Un fatto interessante emerso è che AlphaZero ha inizialmente appreso da sé aperture note e praticate tutt'ora al giorno d'oggi dai normali giocatori di scacchi, ma durante il processo di addestramento ha spesso abbandonato queste varianti perché ha trovato diversi modi per confutarle! Potrebbe essere un suggerimento del motore riguardo la validità delle aperture comunemente utilizzate dagli scacchisti: secondo lui non sono poi così corrette come generalmente si pensa.

3.2.4 Stile e strategia di gioco

In questo paragrafo, analizziamo lo stile di gioco dell'auto-didatta AlphaZero, basandoci sulle partite giocate e sulle analisi risultanti. Questi concetti possono essere riassunti in una sorta di decalogo, che il motore ha formulato da sé e che segue scrupolosamente per raggiungere la vittoria finale.

1. Attacco al re avversario

È l'obiettivo per antonomasia del gioco, e anche per AlphaZero sembra essere un concetto fondamentale. Il suo stile di gioco è molto aggressivo e il lato dell'arrocco del re avversario è senza dubbio il principale punto focale dove convogliare le proprie forze durante una partita.

2. Sicurezza del proprio re

Poiché il re è il pezzo più preso di mira durante la partita, l'organizzazione difensiva e la protezione del proprio monarca rivestono un ruolo importante. Questo può essere ottenuto, per esempio, tramite l'arrocco e circondando il re di propri pezzi.

3. Controllo del centro

AlphaZero si assicura dapprima che la situazione centrale sia stabile prima di indebolire la propria struttura per aprire linee contro il re avversario. Egli lancia molti attacchi sulle ali, ma lo fa sempre con una solida posizione centrale.

4. Niente paura quando si tratta di sacrificare

Il motore non teme affatto i sacrifici di materiale, purché siano eseguiti per aprire colonne o diagonali nei confronti del re avversario. La caratteristica strabiliante dei suoi sacrifici è che sono sacrifici a lungo termine: non c'è un compenso immediato come uno scacco matto o una sequenza tattica che permette di recuperare qualche pezzo.

5. In apertura, i pedoni non contano

All'inizio di una partita di scacchi, AlphaZero non esita a sacrificare uno o due pedoni per aprire linee o diagonali contro il re avversario. Limitare il pezzo più importante dell'avversario costituisce il fulcro del calcolo del super-computer.

6. Sfruttare al massimo la coppia degli alfieri

Come ben descritto dalla letteratura scacchistica, conservare entrambi i propri alfieri (quello delle case chiare e l'altro delle case scure) può conferire un notevole vantaggio, specialmente in una posizione sgombera di pedoni e abbastanza aperta. È ben noto che la coppia degli alfieri valga molto di più in confronto alla coppia di cavalli, nonostante presi singolarmente il loro valore individuale sia simile. AlphaZero dimostra abilmente come sfruttare la sua coppia di alfieri come una formidabile forza d'attacco contro (ovviamente) il re avversario.

7. Cercare ottimi avamposti per i propri cavalli

Il cavallo è un pezzo peculiare e spesso può rivelarsi fatale per i pezzi avversari, dato che è l'unico pezzo che ha la capacità di "saltare" i pezzi presenti sulla scacchiera (vedi Appendice A). Tuttavia, è essenziale sfruttare a pieno il potenziale di questo pezzo, e per far ciò è fondamentale trovare una casella dalla quale può controllare zone nevralgiche della struttura nemica (possibilmente vicino al centro, per non limitare il suo raggio d'azione). Queste posizioni, nella letteratura scacchistica, vengono comunemente chiamate *avamposti*. AlphaZero è maestro nel riconoscere gli avamposti ideali per i suoi cavalli e non esita a sacrificare materiale per guadagnare tempo e spostarli in queste posizioni strategiche.

8. Neutralizzare i pezzi avversari

Forse è il tratto più strabiliante e incisivo del mostro scacchistico progettato da DeepMind. In molte partite, specialmente contro Stickfish, è possibile osservare come AlphaZero, sempre tramite sacrifici, riesca a imprigionare, bloccare, neutralizzare i pezzi dell'avversario, mantenendoli confinati nelle loro case di partenza fino anche alla mossa 40, senza permettergli di entrare nel vivo del gioco. In questo modo, AlphaZero sfrutta la superiorità numerica dei propri pezzi attivi per circondare e mettere sotto pressione il re avversario. Non è tanto il numero totale dei pezzi sulla scacchiera a fare la differenza, ma quanti di essi contribuiscono all'attacco!

9. Non tutte le colonne sono uguali

AlphaZero non ha paura di ritardare l'occupazione delle colonne aperte con le proprie torri se pensa di poter aprire una colonna contro il re nemico altrove sulla scacchiera.

10. Rompere la coordinazione dei pezzi avversari

Il super-motore è sempre attento alla possibilità di passare a un assalto sull'ala di re.[5] Se i pezzi dell'avversario perdono coordinazione, cercherà di iniziare le operazioni sull'ala di re il più rapidamente possibile.

3.3 Introduzione alle reti neurali

Le *Reti Neurali Artificiali* (ANN) sono alla base della struttura del Deep Learning. Sono delle strutture estremamente versatili, potenti e scalabili, rendendole ideali per affrontare compiti di Machine Learning complessi e di grandi dimensioni. Per esempio, le ANN sono utilizzate per classificare miliardi di immagini (come in Google Images), supportare i servizi di riconoscimento vocale (come Siri di Apple) e consigliare i video migliori da guardare a centinaia di milioni di utenti ogni giorno (come su YouTube).

Sorprendentemente, le Reti Neurali Artificiali (ANN) esistono da tempo: furono introdotte nel 1943 dal neurofisiologo Warren McCulloch e dal matematico Walter Pitts. Nel loro storico articolo "A Logical Calculus of Ideas Immanent in Nervous Activity", presentarono un modello computazionale semplificato di come i neuroni biologici potrebbero lavorare insieme nel cervello degli animali per eseguire calcoli complessi usando la logica proposizionale. Questo fu il primo esempio di architettura di rete neurale artificiale.

Oggi l'interesse per le ANN è in forte crescita e sembra destinata a perdurare, grazie soprattutto a vari motivi: la disponibilità di grandi quantità di dati per l'addestramento, il notevole aumento della potenza di calcolo (in parte grazie alle GPU) e i continui miglioramenti negli algoritmi di apprendimento.

3.3.1 Il neurone biologico

Come suggerisce il nome stesso, le reti neurali prendono spunto dalla cellula biologica presente nella corteccia cerebrale degli animali: il neurone, appunto. Questo è costituito da un corpo cellulare che contiene il *nucleo*, molte estensioni ramificate attorno ad esso, chiamate *dendriti*, e un'estensione molto lunga chiamata *assone*. La lunghezza di quest'ultimo può variare da poche volte quella del corpo cellulare

[5] Nel gioco degli scacchi, l'ala di re (in inglese *kingside*) si riferisce alla metà destra della scacchiera (oppure sinistra se si sta giocando con i pezzi neri), e comprende le colonne e, f, g e h. È chiamata così perché ospita il re all'inizio della partita. In maniera analoga, si definisce l'ala di donna (*queenside*) il lato sinistro (destro per il Nero), ossia quello che comprende le colonne a, b, c e d.

fino a decine di migliaia di volte di più. Verso l'estremità, l'assone si divide in molti rami, alla fine dei quali si trovano piccole strutture chiamate *sinapsi*, le quali hanno il compito di instaurare una comunicazione i dendriti (o il corpo cellulare) di altri neuroni limitrofi. Inoltre, i neuroni biologici ricevono impulsi elettrici brevi attraverso queste sinapsi, chiamati *segnali*.

Nonostante il comportamento individuale di ogni neurone sembri piuttosto semplice, essi sono organizzati in una vasta rete di miliardi di neuroni, ciascuno connesso tipicamente a migliaia di altri neuroni. Questa rete vasta e complessa permette di eseguire calcoli altamente complessi, analogamente a come un formicaio complesso può emergere dagli sforzi combinati di semplici formiche.

3.3.2 Il percettrone

Il primo modello di neurone proposto da Warren McCulloch e Walter Pitts, noto come *neurone artificiale*, è un prototipo molto rudimentale e semplice, che presentava uno o più ingressi binari (ossia che potevano assumere soltanto i valori 0 o 1) e un'unica uscita binaria, che veniva attivata soltanto quando più di un certo numero di ingressi era attivo.

Oggi, tuttavia, è utilizzata un'altra struttura, semplice anch'essa, ma con qualche particolarità in più. Stiamo parlando del *percettrone*, struttura inventata da Frank Rosenblatt ne 1957. Si basa su un neurone artificiale leggermente diverso, chiamato *unità logica a soglia* (TLU) o *unità a soglia lineare* (LTU). In questo modello, gli ingressi e l'uscita sono numeri anziché valori binari e ad ognuno di essi è associata un peso più o meno differente. La TLU calcola una somma pesata degli ingressi $x_1, \ldots, x_n$, e l'output può essere 0 o 1 a seconda che questa somma superi o meno un valore di soglia S. In formule:

$$H(z) = \begin{cases} 1 & \text{se} \quad p_1x_1 + p_2x_2 + \cdots + p_nx_n \geq S \\ 0 & \text{se} \quad p_1x_1 + p_2x_2 + \cdots + p_nx_n < S \end{cases} \tag{3.1}$$

Una singola unità logica a soglia (TLU) può essere utilizzata per una semplice classificazione binaria lineare. La funzione che meglio descrive il comportamento di $H(z)$ è la *funzione di Heaviside*, conosciuta anche come *funzione gradino* (spesso indicata con δ_{-1}, vedere Figura 3.3).

Un percettrone è costituito da un singolo livello di tanti TLU, ciascuno connesso a tutti gli ingressi. Quando ogni neurone di uno strato è collegato a ogni neurone dello strato precedente (detti *neuroni di input*), si parla di *strato completamente*

connesso o *strato denso*. Inoltre, si aggiunge spesso a questo livello un neurone di bias extra ($x_0 = 1$), che produce costantemente il valore 1.

3.3.3 Come addestrare il cervello di una macchina

Frank Rosenblatt ideò un metodo per permettere al suo modello di addestrarsi e migliorarsi autonomamente, ispirandosi alla cosiddetta *regola di Hebb* (o di *apprendimento hebbiano*). Lo psicologo canadese Donald Hebb, intorno alla metà del secolo scorso, propose che se l'attività dell'assone di un neurone A è sufficientemente vicina da eccitare ripetutamente e in modo consistente un altro neurone B, ciò può innescare cambiamenti metabolici e un processo di crescita in entrambi i neuroni. Questo meccanismo è riassunto alla perfezione da Siegrid Löwel con la frase: «Cells that fire together, wire together».

I percettroni vengono addestrati usando una variante di questa regola, che tiene conto degli errori della rete. Durante l'addestramento, il percettrone effettua delle previsioni, e per ogni neurone di output che produce una previsione errata, vengono aggiornati i pesi delle connessioni dagli ingressi che avrebbero dovuto contribuire alla previsione corretta. Questo processo è descritto dall'Equazione 3.2:

$$w_{i,j}^{\text{next}} = w_{i,j} + \eta \left(y_j - \hat{y}_j\right) x_i \tag{3.2}$$

dove:

- $w_{i,j}$ rappresenta il peso di connessione tra il neurone di input i e il neurone di output j,
- x_i è il valore dell'ingresso i dell'attuale istanza di addestramento,
- $\hat{y}_j$ indica l'output del neurone di output j per l'attuale istanza di addestramento,
- y_j è l'output desiderato del neurone di output j per l'attuale istanza di addestramento,
- η rappresenta il tasso di apprendimento.

Poiché il percettrone decide il confine decisionale di ogni neurone di output in maniera lineare, non può apprendere modelli complessi. Tuttavia, molte limitazioni dei percettroni possono essere inserendo più livello. La rete neurale risultante viene chiamata *Multi-Layer Perceptron* (MLP). Esso è costituito da:

1. un livello di input (*input layer*),

2. più livelli nascosti (*hidden layewrs*),

3. un ultimo livello finale (*output layer*).

Gli strati vicini allo strato di input sono solitamente chiamati strati inferiori, mentre quelli vicini agli output sono chiamati strati superiori. Quando una rete neurale artificiale (ANN) contiene una serie profonda di strati nascosti, viene chiamata *rete neurale profonda* (DNN).

3.3.4 La funzione di attivazione sigmoidale

Il modello di neurone che abbiamo esaminato presenta alcune limitazioni dovute alla natura della funzione gradino utilizzata come criterio decisionale. Il problema principale è legato al salto repentino tra i valori intorno allo zero. La derivata della funzione gradino è la delta di Dirac (indicata con δ), una distribuzione generalizzata che è infinitamente piccola ovunque tranne che in $x = 0$, dove è infinitamente grande. Questa discontinuità nell'origine può causare instabilità e problemi nel processo di apprendimento del neurone.

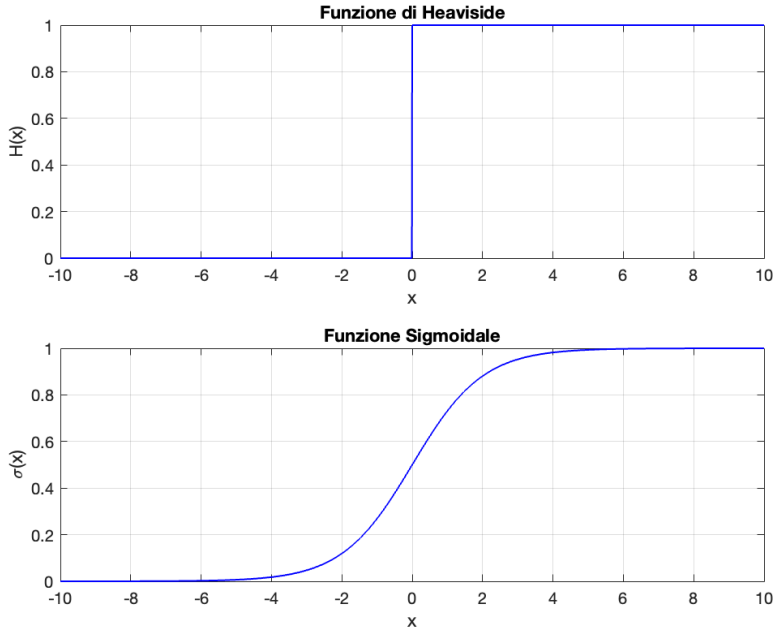


Figura 3.3: La funzione gradino e la funzione sigmoidale a confronto.

Per ovviare a questo inconveniente, si è ricorso all'utilizzo del *neurone sigmoidale*, il cui output è ben descritto dalla *funzione sigmoidale*. Tale funzione ha la proprietà di variare gradualmente intorno a $x = 0$, offrendo un gradiente continuo e ben definito

che è essenziale per il funzionamento efficace dell'algoritmo di retro-propagazione dell'errore.

Inoltre, esaminando la derivata della funzione sigmoidale (Equazione 3.3), si può notare che essa corrisponde a una distribuzione gaussiana, caratterizzata da una forma a campana. Questa proprietà rende la funzione sigmoidale più gestibile e stabile rispetto alla delta di Dirac, facilitando il processo di ottimizzazione dei pesi del neurone durante l'addestramento del modello.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad \sigma'(x) = \frac{e^x}{(e^x + 1)^2} \tag{3.3}$$

3.4 Uno scontro tra titani

Nel dicembre del 2017 si è svolta una sfida epica nella storia degli scacchi computerizzati. Da un lato, Stockfish versione 8, uno dei motori scacchistici che domininava il panorama globale. Dall'altro, il suo sfidante, niente meno che AlphaZero, il rivoluzionario motore sviluppato da DeepMind. È stata una sfida tra presente e futuro, tra vecchio stile e nuova generazione, tra potenza di calcolo bruta e ricerca tramite reti neurali (tra capitolo 2 e i primi paragrafi del capitolo 3 di questa tesi).

Nelle 100 partite giocate, AlphaZero ha sorprendentemente dominato il match, vincendone 28 e pareggiando le restanti. Il suo stile di gioco e l'efficacia delle sue mosse ha spinto la comunità open source di Stockfish a integrare le reti neurali anche nella propria architettura. Di conseguenza, anche tutti gli altri motori scacchistici hanno adottato questa novità.

In questo paragrafo analizziamo una delle partite più iconiche, denominata *Immortal Zugzwang Game*. Capiremo presto il motivo di tale nome. In Figura 3.4 è possibile osservare la posizione di questa partita alla mossa 33.

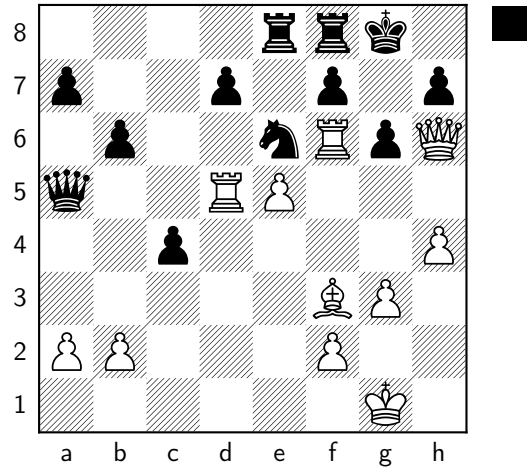


Figura 3.4: AlphaZero vs Stockfish, mossa 33. Immortal Zugzwang Game. Dicembre 2017.

Come illustrato nel Paragrafo 3.2.4, AlphaZero applica con rigore i principi delle reti neurali: non ha esitato, infatti, a sacrificare un pedone già alla sesta mossa per aprire linee verso i pezzi avversari. Ora il prossimo passo sembra essere quello di intrappolare i pezzi neri in una sorta di gabbia, cosa che in parte è riuscita se si osservano la poca mobilità del re nero e delle due torri ancora fermi sulla traversa di partenza. Soffermiamoci ora sulla posizione raggiunta una dozzina di mosse dopo (Figura 3.5):

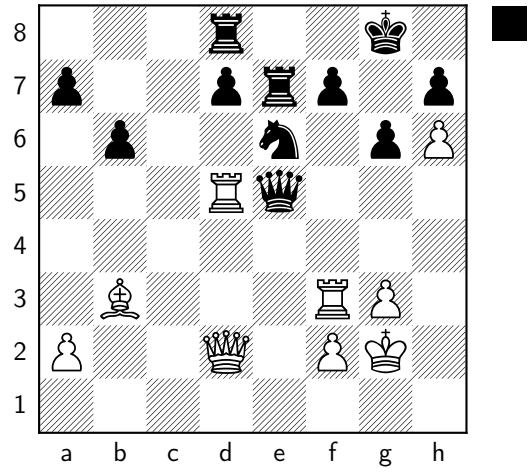


Figura 3.5: AlphaZero vs Stockfish, mossa 45. Immortal Zugzwang Game. Dicembre 2017.

Siamo ad un punto di svolta: AlphaZero ha sacrificato il suo prezioso pedone centrale in e5 per adescare la donna avversaria in una sorta di trappola. Dopo la mossa

45 del Bianco, il Nero è costretto a ritirare la sua regina nell'angolo in h8. Da questo momento in poi, è una questione di maestria per AlphaZero imprigionare definitivamente il pezzo avversario più potente.

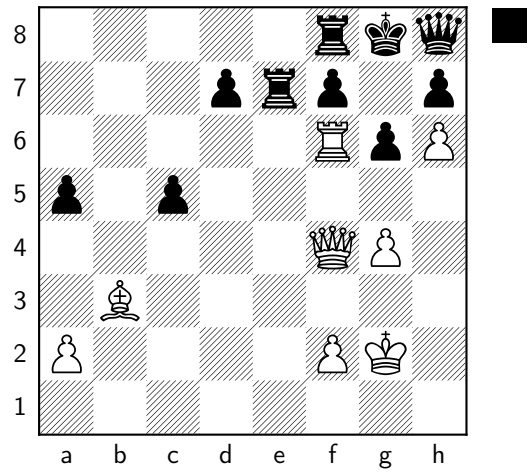


Figura 3.6: AlphaZero vs Stockfish, mossa 51. Immortal Zugzwang Game. Dicembre 2017.

Alla mossa 51 (Figura 3.6) si raggiunge la posizione cruciale di tutta la partita. Il Nero non ha più mosse utili: i suoi pezzi sono incastrati e si trovano in posizioni scomode. Sarebbe perfetto se si potesse congelare la posizione e far scorrere il tempo, ma il gioco degli scacchi non lo consente. Il Nero è in *zugzwang*: è costretto a muovere, e qualsiasi mossa compia peggiorerà inevitabilmente la sua situazione.

Stockfish ha resistito con onore fino alla sessantesima mossa, ma alla fine non ha potuto nulla contro la superiorità del suo avversario.

Capitolo 4

Analisi di ChessMind Pro

In questo capitolo viene presentato "ChessMind Pro", un avanzato motore di scacchi sviluppato nell'ambito di questa tesi, realizzato mediante l'integrazione di tecnologie di intelligenza artificiale e reti neurali. A differenza dei motori tradizionali, ChessMind Pro non è stato programmato con le regole del gioco. Invece, apprende quali mosse eseguire in base alla posizione attuale della scacchiera, analizzando un vasto dataset di partite giocate dai grandi campioni del presente e del passato. Questo approccio innovativo permette al motore di assimilare le strategie e lo stile dei maestri, offrendo una visione unica delle capacità di apprendimento dell'AI e del suo potenziale applicativo nel mondo degli scacchi.

4.1 Introduzione al ML

Introduciamo dapprima il concetto di *machine learning* (ML), o apprendimento automatico, una branca dell'intelligenza artificiale che è alla base della struttura di ChessMind Pro. Il ML permette ai computer di apprendere e migliorare autonomamente dalle esperienze passate, senza necessità di essere programmati esplicitamente per ogni compito specifico. Questo approccio si basa sull'analisi di grandi quantità di dati che vengono utilizzati per addestrare degli algoritmi in grado di identificare autonomamente le regole e i pattern sottostanti.

Prendiamo in considerazione, ad esempio, il monitoraggio delle prestazioni di un atleta nel corso di un'attività sportiva. Utilizzando sensori indossabili che monitorano parametri come la frequenza cardiaca, la posizione e la velocità, possiamo raccogliere dati importanti sull'attività fisiche dell'atleta. Questi dati grezzi vengono poi etichettati con le attività corrispondenti, come "camminare", "correre" o "sedersi". Il machine learning entra in gioco quando utilizziamo questi dati etichettati per addestrare un determinato modello. Tale modello impara a riconoscere le

caratteristiche distintive di ogni attività (ad esempio, un aumento della frequenza cardiaca e della velocità può indicare che una persona sta correndo), in modo tale da renderlo capace di prevedere l'attività in corso semplicemente analizzando i dati provenienti da questi sensori.

4.1.1 Che cos'è il *Learning*

Gli algoritmi di machine learning fanno riferimento al concetto di apprendimento (di *learning*, appunto), che però non deve essere confuso con l'apprendimento umano o animale. Questi algoritmi non "imparano" nel senso tradizionale del termine; piuttosto, si tratta di processi che stimano parametri all'interno di un modello matematico statistico.

L'obiettivo del learning, infatti, è ottimizzare una funzione che rappresenta il modello, ovvero un sistema che stabilisce una relazione tra input e output. Pertanto, quando si riferisce ad una stima dei parametri, il richiamo è semplicemente all'ottimizzazione di una funzione, sia essa analitica (derivate, ...) o numerica (metodi di discesa di gradiente, ...).

I passi che segue un dispositivo che sfrutta il ML, solitamente, sono i seguenti:

1. Raccolta dei dati

Questa è la prima fase del processo di apprendimento. Può avvenire tramite sensori o implementazione diretta da parte di un operatore umano. Ad esempio, nei sistemi di monitoraggio, sono installati numerosi sensori che possono raccogliere informazioni da dispositivi indossabili; in altri contesti, i dati possono essere raccolti manualmente o provenire da banche dati apposite.

Nella maggior parte dei casi, i dati raccolti vengono sottoposti ad una fase di preprocessing, fondamentale per le operazioni di pulizia dei dati spuri, come la rimozione o la correzione di valori mancanti o errati. Inoltre, i dati vengono normalizzati, ovvero sottoposti ad operazioni che prevedono di uniformare la scala dei valori di input. Questo può avvenire tramite un processo di standardizzazione, in cui i dati vengono modificati in modo tale da avere, per facilitare l'addestramento del modello, una media centrata su 0 e una varianza pari a 1.

2. Etichettatura dei dati

È fondamentale distinguere tra due approcci distinti di learning: *supervised* (o supervisionati) o *unsupervised* (non supervisionati). In questi ultimi i dati non sono etichettati, cioè non esiste un output predefinito associato agli input. Questo rende

il processo più complesso, poiché il modello ha a disposizione solo i dati in input e deve individuarvi autonomamente le relazioni sottostanti senza avere a disposizione alcuna informazione a priori riguardo il risultato da predire.

Nel caso del supervised learning, l'obiettivo del modello è quello di costruire una funzione in grado di prevedere l'etichetta di output a partire dai dati forniti. Pertanto, il modello viene addestrato su un dataset con etichette conosciute, simulando un processo di "learning with teacher", in cui il sistema conosce sia i dati di input che quelli di output ed è guidato nella costruzione della funzione di valutazione, che gli permette di ottimizzare l'associazione tra i dati e le etichette.

3. Addestramento (*training*)

Il *training* rappresenta uno degli step più importanti nel processo di apprendimento di un modello di ML. In questa fase, il modello viene addestrato utilizzando i dati provenienti dal *training set*; successivamente, le sue performance vengono valutate su un *test set*.[1] Durante il training, i parametri interni della rete neurale vengono ottimizzati in modo da minimizzare il più possibile l'errore tra l'output predetto dal modello e il valore desiderato.

In sostanza, questa è la fase in cui il modello realizza una funzione che lega i valori di input ai valori di output. Ciò permette alla rete di adattarsi ai dati in ingresso e le permette di generalizzare efficientemente, anche quando viene esposta a nuovi dati non precedentemente analizzati.

4. Previsione

Una volta completato l'addestramento, il modello è ora pronto per effettuare previsioni (*prediction* o *inference*) sui nuovi dati forniti. Ed è proprio grazie ai parametri precedentemente ottimizzati che il modello sarà in grado di ottenere predizioni il più vicino possibile al risultato desiderato.

4.1.2 Tipologie di problemi nel ML

La capacità di riconoscere schemi e automazioni rende gli algoritmi di machine learning lo strumento ideale per svolgere molte operazioni complesse e risolvere problemi ricorrenti nella vita di tutti i giorni. Molte attività, infatti, possono essere eseguite tranquillamente da una macchina ben addestrata, come il riconoscimento

[1] Solitamente, il training e il test set derivano dallo stesso dataset, il quale viene suddiviso in modo tale che la maggior parte dei dati sia utilizzata per il training (solitamente l'80%) e la restante (20%) per il test.

vocale del testo, il filtraggio dello spam nelle e-mail, la visione artificiale, l'analisi medica, e molto altro ...

Tuttavia, nell'ambito del ML, queste attività non sono tutte uguali, ma possono essere suddivise in tre principali categorie di problemi, a seconda dell'output desiderato:

1. Problemi di classificazione: l'obiettivo è assegnare ogni oggetto a una specifica classe tra un insieme predefinito di classi; generalmente questi problemi vengono affrontati con un approccio supervisionato.

2. Problemi di regressione: condividono in larga scala la fase di generazione e suddivisione delle etichette della classificazione; tuttavia, in uscita non si hanno più dei numeri discreti, ma dei valori appartenenti all'insieme dei numeri reali. Un esempio di problema di regressione è la previsione del mercato azionario o della variazione di temperatura in una determinata località.

3. Problemi di clusting: tipicamente si basano su un approccio non supervisionato, dato che non si dispongono di etichette che caratterizzano i dati.

4.1.3 Uno sguardo ai diversi algoritmi

A seconda dei compiti che gli vengono assegnati, un modello di ML deve essere in grado di rispondere, con accuratezza e precisione, ad ogni singola richiesta. Per far fronte a questa esigenza, esistono tra i più svariati algoritmi di ML, ognuno specializzato in un task ben definito. In questo modo, a seconda del problema da risolvere, si hanno sempre a disposizione il metodo e lo strumento migliore per risolverlo.

Ad esempio, l'utilizzo di *shallow neural networks* (reti neurali superficiali) è un approccio basato su una rete neurale relativamente semplice (composta da uno strato di input, uno o due strati nascosti e un ultimo strato di output) che permette di risolvere problemi di ML di piccola complessità.

Oltre alle reti neurali, esistono approcci più tradizionali che non si basano sui neuroni. Tra questi troviamo la regressione lineare, il cui obiettivo è quello di cercare la retta che maggiormente si adatta alla distribuzione dei dati, le *support vector machines* (SVM) e gli alberi decisionali.

Tuttavia, non entreremo nel dettaglio di queste tecniche, poiché un approfondimento non rientrerebbe tra gli obiettivi principali di questa tesi.

4.2 L'addestramento di ChessMind Pro

4.2.1 La codifica della scacchiera

Ritornando al nostro motore di scacchi, in questo paragrafo descriveremo i vari passaggi effettuati per addestrare il modello. Il primo passo è stato quello di definire una notazione in grado di rappresentare accuratamente lo stato attuale della scacchiera. Questa include non solo la disposizione di ogni pezzo, ma anche informazioni aggiuntive rilevanti, quali:

- il turno di gioco
- le possibilità di arrocco per ciascun giocatore
- dettagli sull'ultima mossa effettuata, utile (per esempio) per eseguire la *presa en passant*[2]
- il conteggio delle mosse per determinare un'eventuale patta per la regola delle 50 mosse.

È importante sottolineare come siano state volontariamente escluse alcune informazioni cruciali di una partita di scacchi, come il numero consecutivo di semimosse senza cattura o senza spostare pedoni (utile nei finali per decretare la patta) e il conteggio della mossa rispetto all'inizio della partita. Tuttavia, queste informazioni non sono considerate necessarie per la costruzione della nostra architettura.

Per garantire una corretta interpretazione del dataset utilizzato per addestrare il modello, è stato fondamentale convertire ogni posizione della scacchiera in una sequenza di numeri, ciascuno dei quali rappresenta una specifica informazione relativa allo stato del gioco. Questo processo traduce le informazioni in un formato comprensibile per il motore e le rende utilizzabili dall'algoritmo di intelligenza artificiale.

Per descrivere lo stato di ogni casella della scacchiera è stato relativamente semplice ricorrere alla rappresentazione ordinale. La scacchiera, infatti, può essere rappresentata come un array di 64 elementi, dove ogni casella è identificata da un indice univoco calcolato a partire dalla riga e dalla colonna corrispondenti, secondo la formula:

$$i = i_c - 1 + 8 \cdot (i_r - 1) \tag{4.1}$$

[2]La *presa en passant* è una mossa speciale del gioco degli scacchi, che può essere effettuata da un pedone soltanto in determinate condizioni. Per maggiori informazioni consultare Appendice A.

con i_c e i_r valori compresi tra 0 e 7. Ognuna di queste 64 caselle può trovarsi in uno di tredici stati differenti: vuota, oppure occupata da uno dei sei pezzi bianchi o uno dei sei pezzi neri (vedi Figura 4.1). In particolare, sono stati attribuiti i seguenti valori: 0 (vuota), 1 (re bianco), 2 (regina bianca), 3 (alfiere bianco), 4 (cavallo bianco), 5 (torre bianca), 6 (pedone bianco), 7 (re nero), 8 (regina nera), 9 (alfiere nero), 10 (cavallo nero), 11 (torre nera) e 12 (pedone nero).

56	57	58	59	60	61	62	63
48	49	50	51	52	53	54	55
40	41	42	43	44	45	46	47
32	33	34	35	36	37	38	39
24	25	26	27	28	29	30	31
16	17	18	19	20	21	22	23
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7

Figura 4.1: Indici delle caselle della scacchiera.

È stato doveroso, successivamente, utilizzare un numero per indicare il giocatore a cui spetta la mossa: in particolare indichiamo con 0 il turno del Bianco, mentre con 6 il turno del Nero.

Successivamente, sono stati indispensabili altri 4 numeri per indicare rispettivamente le possibilità dei seguenti arrocchi:

- 0 $\longrightarrow$ indica che il Bianco può effettuare l'arrocco corto,
- 1 $\longrightarrow$ indica che il Bianco può effettuare l'arrocco lungo,
- 6 $\longrightarrow$ indica che il Nero può effettuare l'arrocco corto,
- 7 $\longrightarrow$ indica che il Nero può effettuare l'arrocco lungo.

I seguenti valori sono sostituiti con -1 qualora non si verifichi una specifica condizione.

Infine, è stato necessario un ulteriore numero per indicare la possibilità di catturare un pedone en passant: se non vi è questa possibilità, viene inserito di default il valore -1; altrimenti andrebbe inserito il numero corrispondente all'indice della casa nella quale può avvenire la cattura (quindi un numero tra 0 e 63).

Alla fine di tutti questi ragionamenti, siamo arrivati ad ottenere in totale 70 valori, fondamentali per indicare univocamente una determinata posizione di scacchi. Tuttavia, sono necessari ulteriori 7 numeri per designare la mossa che deve essere eseguita dal motore, in particolare:

1. indica il numero corrispondente al pezzo che viene mosso
2. è l'indice della casa di partenza del pezzo
3. è l'indice della casa di arrivo sempre dello stesso pezzo
4. sta ad indicare l'eventuale pezzo a cui è promosso un pedone giunto all'ultima traversa (è -1 di default)
5. sempre in caso di promozione, è l'indice della casa di promozione (è 0 di default)
6. in caso di presa en passant, indica la casa da liberare (è -1 di default)
7. sta ad indicare se è avvenuta o meno la presa en passant (è -1 di default, è 0 se è avvenuta la cattura)

$$\underbrace{64}_{\text{scacchiera}} + \underbrace{1}_{\text{turno}} + \underbrace{4}_{\text{arrocchi}} + \underbrace{1}_{\text{en passant}} + \underbrace{7}_{\text{mossa}} = 77 \tag{4.2}$$

4.2.2 La preparazione del dataset

Per lo step successivo, il compito è stato quello di reperire la materia prima necessaria per l'addestramento di ChessMind Pro: era indispensabile trovare un numero considerevole di partite di scacchi, con le quali il motore avrebbe potuto esercitarsi nell'analisi delle mosse e nell'apprendere quelle corrette da eseguire.

A tale scopo, è stato di grande aiuto il sito www.pgnmentor.com, che offre una vasta collezione di partite giocate dai grandi campioni della storia. In totale, il sito mette a disposizione oltre 500.000 mila partite facilmente consultabili e scaricabili. Facendo un rapido conto, supponendo che ogni partita duri in media 40 mosse, sarebbe teoricamente possibile esaminare un numero di mosse pari a 20 milioni, e quindi si avrebbe a disposizione un dataset molto corposo e completo.

Successivamente, tutte queste partite sono state facilmente elaborata utilizzando la notazione PGN degli scacchi, un formato standard che permette di registrare, conservare e gestire tutte le informazioni relative allo svolgimento di una partita. Ogni mossa è stata codificata secondo le regole dettate nel paragrafo precedente e, procedendo manualmente per ogni partita, è stato costruito il dataset. Quando

il nome di un grande campione era presente accanto al colore “White” o “Black” (quindi aveva giocato rispettivamente con i pezzi bianchi o con i pezzi neri), venivano registrate le mosse effettuate (ovvero gli ultimi 7 numeri della Equazione (4.2)), mentre la posizione precedente a quella mossa veniva memorizzata nei primi 70 numeri, ossia quelli che hanno lo scopo di rappresentare la scacchiera in quel preciso momento.

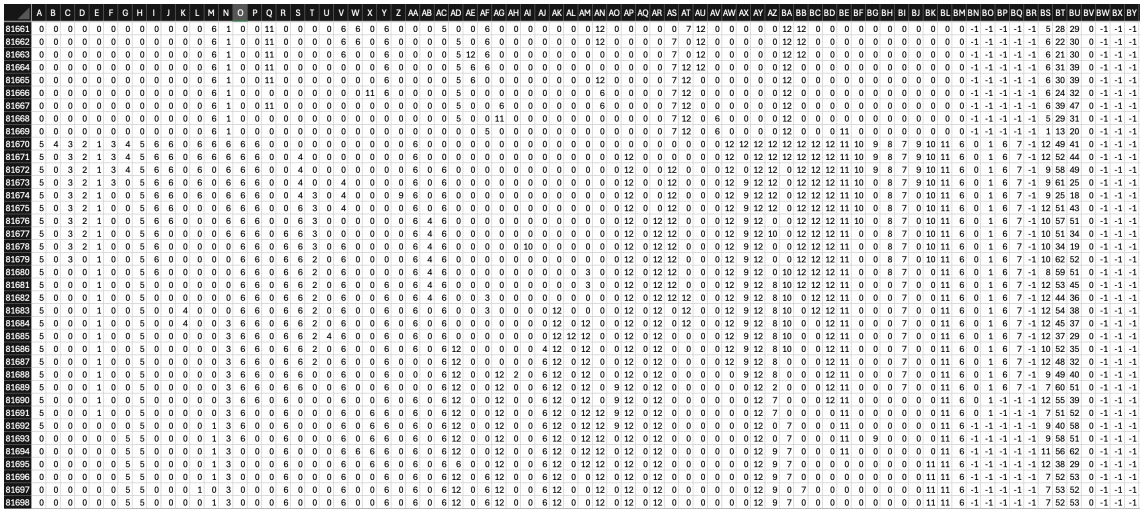


Figura 4.2: Esempio di dataset utilizzato per l’addestramento di ChessMind Pro, estratto da un foglio di calcolo Excel. Ogni riga rappresenta una singola disposizione della scacchiera, con i valori numerici che indicano i pezzi, la mossa da eseguire e altre informazioni utili.

Il numero di righe effettivamente utilizzate per l’addestramento è stato ridotto a circa 200.000, poiché, a causa delle limitate capacità computazionali dell’hardware disponibile, il programma iniziava a rallentare le proprie prestazioni. Di conseguenza, è stato necessario trovare un trade-off accettabile tra la complessità dei dati di input e l’efficienza del processo di addestramento, in modo tale da garantire allo stesso tempo buone performance senza compromettere eccessivamente la velocità del sistema.

4.2.3 Il training del modello

L’addestramento del motore di scacchi è stato eseguito su un MacBook Pro equipaggiato con chip “Apple M1”, il quale integra una CPU 8 core (composta da 4 core ad alte prestazioni e 4 core ad alta efficienza) una GPU integrata a 8 core e una RAM pari a 8 GB. Inoltre, il dispositivo è supportato da un Neural Engine a 16 core,

progettato per accelerare il calcolo delle reti neurali e per renderlo particolarmente efficiente nelle attività di machine learning.

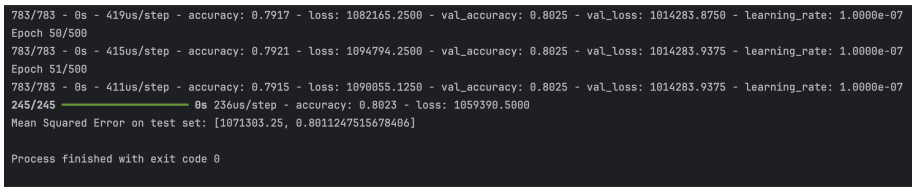


Figura 4.3: Esempio di output durante l'addestramento di ChessMind Pro, il quale mostra informazioni utili sul progresso del training.

In questa fase, è stato necessario utilizzare una *funzione di loss* per quantificare l'errore commesso dal modello durante le sue previsioni. Nel nostro caso, si è optato per l'errore quadratico medio (*mean square value*, MSE), una funzione che calcola la media statistica delle differenze al quadrato tra i valori predetti e quelli desiderati. Inoltre, tale funzione è stata ulteriormente modificata per restituire dei valori particolarmente elevati nel caso in il modello preveda una mossa illegale, come ignorare le regole di movimento di un pezzo o non spostare il re quando è sotto scacco.

L'addestramento è proseguito con tutti i dati precedentemente raccolti finché il valore della funzione di perdita non si è ridotto significativamente rispetto all'errore calcolato ad inizio addestramento. Ciò avviene, come abbiamo accennato in maniera più generale, perché ad ogni iterazione di addestramento (in gergo di ML "epoca") la funzione di loss guida l'ottimizzare dei parametri interni del modello, in modo da ridurre il più possibile l'errore calcolato.

Grazie a queste accortezze, ChessMind Pro è in grado di apprendere e adattarsi anche a nuove posizioni che emergono durante una partita, permettendogli di elaborare delle mosse originali che non erano presenti nel suo dataset di addestramento. Questo approccio consente al motore di scacchi di rispondere con successo ad a una vasta gamma di configurazioni di gioco e di mantenere un alto livello di precisione e competenza in ogni fase della partita.

4.3 Caratteristiche del motore

4.3.1 Ambiente di sviluppo e librerie importate

Il motore di scacchi ChessMind Pro è stato sviluppato utilizzando Python, un linguaggio di programmazione ampiamente diffuso e particolarmente adatto per applicazioni di intelligenza artificiale e machine learning. Per lo sviluppo del progetto sono state utilizzate diverse librerie, ognuna specializzata in ambiti come l'IA, l'ela-

borazione dei dati e la gestione delle partite di scacchi. Di seguito, sono descritte le librerie più importanti utilizzate e il loro ruolo all'interno del programma.

- **chess**: la libreria "python-chess" è stata molto importante per la gestione e la manipolazione delle partite di scacchi del dataset. Fornisce tutte le classi e i metodi per creare, visualizzare e analizzare ogni mossa, gestire ogni posizione sulla scacchiera scacchiera e verificare le mosse illegali.

- **numpy**: Libreria utilizzata per la gestione efficiente delle operazioni matematiche e delle strutture dati multidimensionali.

- **tensorflow**: TensorFlow è una delle librerie più popolari per il machine learning e le reti neurali. In ChessMind Pro, viene utilizzato per addestrare il modello di rete neurale.

- **keras**: Keras è un'API di alto livello per la costruzione e l'addestramento di reti neurali che si integra con TensorFlow. Esso semplifica la creazione della rete neurale, permettendo di definire facilmente gli strati del modello e di gestire la fase di training in maniera più efficiente.

- **scikit-learn**: Fondamentale per l'implementazione di algoritmi di machine learning, utilizzata principalmente per il pre-processing dei dati e per la validazione del modello di AI.

- **torch**: "PyTorch" è una libreria avanzata per il machine learning, in particolare per le reti neurali. Viene impiegata in alcune versioni sperimentali di ChessMind Pro per testare la sua flessibilità in confronto a TensorFlow, data la sua capacità di eseguire calcoli in maniera dinamica e l'efficace gestione delle reti neurali complesse.

- **h5py**: Permette la gestione di file in formato "HDF5", che vengono utilizzati per il salvataggio e il caricamento di grandi quantità di dati. Nel nostro caso, viene utilizzata per memorizzare i pesi e i parametri delle reti neurali addestrate.

4.3.2 Rete neurale impiegata

La rete neurale impiegata nel modello è una MLP (*Multi-Layer Perceptron*) densa, costituita in particolare da 3 hidden layer (Tabella 4.1).

In generale, una rete MLP densa è una rete neurale feed-forward[3] in cui ogni neurone di uno strato è connesso a tutti i neuroni dello strato successivo. Questo tipo di architettura offre alla rete una visione più completa, poiché incrementa le interazioni tra i neuroni e consente al modello di apprendere informazioni più complesse.

Nell'architettura di ChessMind Pro, per consentire al modello di apprendere relazioni complesse, è stata utilizzata la funzione di attivazione ReLU (*Rectified Linera Unit*) per introdurre non-linearità negli hidden layer e una funzione di attivazione lineare nel layer di output, ottimale per eseguire previsioni numeriche.

Infine, per prevenire il fenomeno dell'overfitting,[4] la rete è stata regolarizzata utilizzando la tecnica di "dropout" al 30% tra i vari hidden layer. Ciò permette di disattivare casualmente un certo numero di neuroni che, durante l'addestramento, non contribuiscono all'apprendimento del modello, ma agiscono semplicemente come dei "passaparola" per i neuroni successivi.

Layer (type)	**Output Shape**	**Param #**
Dense	(None, 64)	4,544
Dense_1	(None, 32)	2,080
Dense_2	(None, 16)	528
Dense_3	(None, 7)	119

Tabella 4.1: Architettura del modello sequenziale MLP adottato, con dettagli sui layer, il numero di neuroni e i parametri da ottimizzare per ciascuno strato.

4.3.3 Codice sorgente

Nel seguente paragrafo è illustrato il codice sorgente di ChessMind Pro, ossia il vero e proprio cuore pulsante dell'innovativo motore di scacchi. Il codice è strutturato in diverse sezioni logiche, ognuna con un ruolo specifico e fondamentale, descritte brevemente nel paragrafo successivo.

[3]Nel campo del ML, una rete neurale feed-forward è una architettura in cui le connessioni tra i nodi non formano cicli. In questo modo, le informazioni fluiscono in una determinata direzione, a partire dai nodi d'ingresso fino a quelli d'uscita. Un esempio di reti di questo tipo sono i percettroni a singolo strato (SLP) e i percettroni a multistrato (MLP).

[4]In statistica e informatica, l'overfitting (o sovradattamento) si verifica quando un modello è troppo complesso rispetto alla quantità di dati disponibili o quando sono presenti troppe variabili rispetto alla dimensione del set di dati. Questo porta il modello ad adattarsi perfettamente ai dati di addestramento, ma potrebbe non funzionare altrettanto bene su dati nuovi o di test.

```
# Step 0: parametri di configurazione
attempts = 500
level_patience = 100
model_file = "chess_mind_pro.keras"

# Step 1: Caricamento dei dati
file_paths = ['anand.csv','kasparov.csv','tal.csv','carlsen.csv','fischer.csv',...]
dataframes = []
for file_path in file_paths:
    df = pd.read_csv(file_path, header=None)
    dataframes.append(df)
dataset_chess = pd.concat(dataframes, ignore_index=True)

# Step 2: Preparazione del dataset
X = dataset_chess.iloc[:, :70].values
y = dataset_chess.iloc[:, 70:].values

# Step 3: Suddivisione del dataset in training e test set
X_train_full, X_test, y_train_full, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train, X_valid, y_train, y_valid = train_test_split(X_train_full, y_train_full, test_size=0.2,
 random_state=42)

# Step 4: Normalizzazione dei dati
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_valid_scaled = scaler.transform(X_valid)
X_test_scaled = scaler.transform(X_test)

# Step 5: Definizione del modello, se non esiste gia'
if os.path.exists(model_file):
    model = keras.models.load_model(model_file)
    print("Model loaded successfully from disk!")
else:
    model = keras.models.Sequential([
        keras.Input(shape=(70,)),
        keras.layers.Dense(64, activation="relu"),
        keras.layers.Dropout(0.3),
        keras.layers.Dense(32, activation="relu"),
        keras.layers.Dropout(0.3),
        keras.layers.Dense(16, activation="relu"),
        keras.layers.Dense(7, activation="linear")
    ])

# Compilazione
model.compile(loss="mean_squared_error", optimizer="adam", metrics=["accuracy"])

model.summary()         # grafica l'architettura utilizzata

# Callbacks per il salvataggio del modello durante l'addestramento
checkpoint = [
    keras.callbacks.ModelCheckpoint(model_file, save_best_only=True),
    keras.callbacks.EarlyStopping(patience=level_patience, restore_best_weights=True),
    keras.callbacks.ReduceLROnPlateau(factor=0.1, patience=10)
]

# Addestramento
history = model.fit(X_train_scaled, y_train, epochs=attempts, validation_data=(X_valid_scaled, y_valid),
                    verbose=2, callbacks=checkpoint)

# Salvataggio
model.save(model_file)

# Step 6: Valutazione del modello
mse_test = model.evaluate(X_test_scaled, y_test)
print(f"Mean Squared Error on test set: {mse_test}")

# Step 7: Grafica i risultati ottenuti
pd.DataFrame(history.history).plot(figsize=(8,5))
plt.grid(True)
plt.show()
```

4.3.4 Fasi del modello neurale

I vari step presenti nel codice sono i seguenti:

1. **Configurazione**. Vengono definiti i parametri principali per ottimizzare il processo di training, come il numero massimo di epoche da eseguire in assenza di miglioramenti nella rete neurale, insieme ad altre informazioni rilevanti, come il percorso per il salvataggio del modello e la directory del dataset.

2. **Preparazione del dataset**. Le partite di scacchi utilizzate sono caricate in dei file in formato `.csv`, il quale è suddiviso in una parte di input (X) e target (Y), dove gli input rappresentano la configurazione della scacchiera e altre informazioni di gioco (ad esempio chi deve muovere, diritti di arrocco, mossa en passant), mentre i target sono le mosse da prevedere.

3. **Suddivisione del dataset**. Fondamentale per garantire che il modello possa essere addestrato in modo efficace. Il dataset viene suddiviso in tre parti: *training set* utilizzato per addestrare il modello, *validation set* per monitorare le prestazioni del modello durante l'addestramento e prevenire l'overfitting, *test set* per valutare le prestazioni del modello su nuovi dati.

4. **Normalizzazione dei dati**: Per migliorare le prestazioni, i dati vengono normalizzati (con la funzione "StandardScaler") sottraendo la media e dividendo per la deviazione standard dei campioni utilizzati.

5. **Definizione e compilazione del modello** La rete neurale viene definita, tramite "Keras", con una struttura sequenziale che prevede più livelli densi (vedi Paragrafo 4.3.2).

6. **Valutazione e visualizzazione dei risultati**. Una volta completato l'addestramento, il modello viene valutato sul test set. In particolare, viene calcolato l'errore quadratico medio (*Mean Squared Error*, MSE) per misurare la precisione del modello nelle previsioni.

Nello step conclusivo, il codice genera dei grafici che rappresentano visivamente le metriche di addestramento, come la perdita e l'accuratezza, utili per comprendere meglio il comportamento del modello nel tempo e identificare tempestivamente eventuali problemi.

Questa struttura modulare e ben organizzata del codice consente di ottenere un motore di scacchi potente, in grado di apprendere dalle mosse dei grandi maestri e di migliorare continuamente tramite l'addestramento le sue capacità analitiche.

4.4 Debolezze di ChessMind Pro

4.4.1 Limitazioni computazionali

I risultati ottenuti hanno registrato un discreto apprendimento del modello dal dataset assegnato e una ragionevole predizione di mosse legali, anche se non sempre sono state quelle ottimali. È evidente che si potrebbero ottenere prestazioni più elevate in termini di accuratezza e generalizzazione, soprattutto utilizzando una infrastruttura di calcolo adeguata in grado di gestire dataset più ampi e un numero maggiore di parametri da ottimizzare.

Inoltre, è interessante evidenziare come architetture di reti neurali più avanzate, come i *Large Foundation Models* basati su *Transformer*, potrebbero essere particolarmente più adatte a risolvere questo tipo di problemi. I Transformer, infatti, eccellono nei compiti di trasformazione di sequenze in sequenze (seq2seq), il che li rende utili in contesti che coinvolgono dati strutturati in sequenze, come il linguaggio naturale e, perché no, giochi complessi come gli scacchi.

L'architettura alla base di molti Transformer è basata su un modello *encoder-decoder*, in cui ogni layer svolge un'operazione ben determinata. Alcuni layer si occupano di codificare i dati di input per renderli più agevoli da elaborare (*word embedding*, *positional encoding*); altri hanno il compito di implementare il modello di attenzione vero e proprio. Il vero punto di forza dei Transformer risiede proprio nel loro meccanismo di attenzione, che consente al modello di focalizzarsi sulle parti della sequenza che contribuiscono maggiormente alla sua comprensione del contesto.

Tuttavia, vale la pena notare che l'addestramento è stato effettivamente completato e il lavoro svolto rappresenta un importante passo avanti nello sviluppo del modello. Pertanto, sono sicuramente presenti tutte le premesse per migliorare le prestazioni del motore di scacchi e, allo stesso tempo, solide fondamenta su cui costruire una macchina altamente competitiva e potenzialmente imbattibile.

4.4.2 Dalla forza nelle aperture alle difficoltà nei finali

Un motore di scacchi come ChessMind Pro, che funziona tramite intelligenza artificiale e apprende dalle partite di grandi campioni, è particolarmente abile nel rispondere alle posizioni d'apertura. Questo è dovuto alla vasta quantità di partite disponibili che documentano le aperture comuni e le varianti strategiche più efficaci all'inizio di ogni partita di scacchi. Grazie a queste informazioni, il motore può analizzare rapidamente le posizioni iniziali e proporre mosse ottimali, replicando con precisione le strategie utilizzate dai migliori giocatori.

Tuttavia, quando si passa al mediogioco e soprattutto ai finali di partita, le prestazioni del motore possono diminuire. Questo avviene perché il numero di posizioni possibili cresce in modo esponenziale, rendendo le diverse configurazioni sulla scacchiera sempre meno prevedibili. Mentre, infatti, tutte le partite di scacchi iniziano con la posizione (la radice dell'albero delle mosse, vedi Figura 2.1), non vale la stessa cosa per le fasi più avanzate della partita, proprio perché il numero di posizioni possibili cresce vertiginosamente. Nei finali, in particolare, le combinazioni di pezzi sono molto specifiche e richiedono una comprensione approfondita delle tecniche di fine partita. La scarsità di dati disponibili per queste situazioni limita la capacità del motore di apprendere e replicare strategie ottimali.

Per ovviare a questo problema, l'integrazione di tablebase nei motori di scacchi risulterebbe fondamentale. ChessMind Pro, tuttavia, non utilizza le tablebase, ma trarrebbe grande beneficio da esse. Senza queste risorse, il motore ha difficoltà ad affrontare i finali più complessi con la stessa precisione delle aperture. Di conseguenza, ChessMind Pro si auspica di finire la partita prima delle 40 mosse, proprio perché il suo dataset è ricco di mosse iniziali, molto spesso anche simili e identiche tra loro, a differenza delle posizioni finali.

Conclusioni

I motori sono perfetti?

Concludiamo quindi questo breve viaggio storico nell'esplorazione dei vari motori di scacchi che si sono susseguiti ed evoluti nel corso del secolo scorso. È opportuno sottolineare che, nonostante questi motori sembrino avvicinarsi alla perfezione, o a uno stile di gioco quasi divino, essi presentano comunque diversi limiti. D'altronde, non può essere diversamente, dato che sono stati creati da esseri umani, tutt'altro che impeccabili. Analizziamo la posizione nella Figura 4.4:

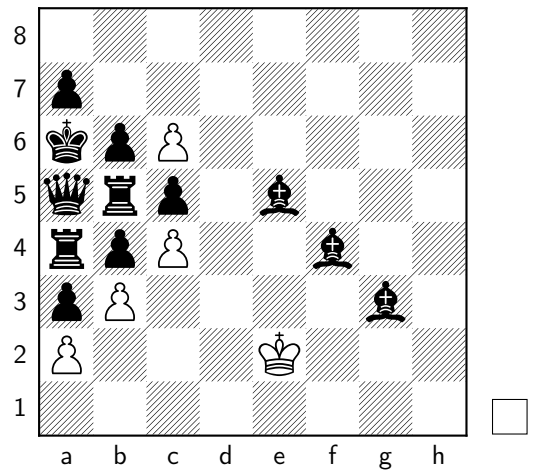


Figura 4.4: Posizione di scacchi, un po' utopistica, che evidenzia una piccola imprecisione nel calcolo computazionale di un motore scacchistico.

La posizione è abbastanza poco probabile che si verifichi nel corso di una partita giocata tra due contendenti (che non siano di comune accordo, ovviamente), data la struttura poco realistica dei pedoni neri e la presenza di ben 3 alfieri camposcuri (è difficile che quando un giocatore va a promozione scelga proprio un pezzo di infimo valore in confronto al grande prestigio della regina). Tuttavia, questa posizione di fantasia è utile per evidenziare alcune imperfezioni negli algoritmi dei motori di scacchi. Pare evidente, anche agli occhi dei meno esperti, che il Nero abbia un

notevole vantaggio materiale: possiede una regina, due torri e tre alfieri camposcuri, mentre il povero monarca dei pezzi bianchi è lasciato alla mercé dei suoi pochi pedoni superstiti.

Qualsiasi motore di scacchi analizzerebbe questa posizione assegnando un grande vantaggio al Nero, data appunto la netta superiorità materiale evidente. Ad esempio, Stockfish 16 NNUE valuta tale posizione con un punteggio di -15.4 a favore del Nero, analizzando fino a una profondità di 61 mosse.[5] Malgrado ciò, la posizione è in realtà perfettamente pari. Il Bianco può mantenere la stabilità della posizione, e quindi la parità, semplicemente muovendo il proprio re e resistendo alla tentazione di spostare uno dei suoi pedoni. Infatti, il monarca nero e la cavalleria pesante a sua disposizione (regina e torri) sono praticamente immobili in una sorta di gabbia. Il secondo giocatore non può fare molti progressi, poiché nessun alfiere controlla le case chiare, impedendo così di dare scacco matto al re avversario.

Questa situazione dimostra che, nonostante la straordinaria potenza dei motori di scacchi, ci sono ancora scenari in cui la comprensione umana delle sfumature posizionali può prevalere, evidenziando le limitazioni intrinseche degli algoritmi di valutazione utilizzati dai motori.

La creatività è a rischio?

Con l'accessibilità dei motori scacchistici a giocatori di ogni livello, tutti hanno potuto beneficiare di questa potente risorsa. Il gioco degli scacchi è diventato sempre più uniforme, con i giocatori che tendono a seguire il "metodo giusto" suggerito dai motori. Questa tendenza solleva preoccupazioni riguardo alla possibile riduzione della creatività individuale. Alcuni sostengono che l'uso dei motori arricchisca il gioco, spingendo i limiti della comprensione umana e permettendo analisi più profonde e sofisticate delle posizioni. Tuttavia, c'è il rischio che i giocatori diventino troppo dipendenti dai motori, preferendo consultare queste risorse piuttosto che sviluppare le proprie capacità analitiche, fondamentali per la crescita personale nel gioco degli scacchi. La vera sfida consiste nel trovare un equilibrio: utilizzare i motori come strumenti educativi per lo studio e l'analisi, ma senza sacrificare la creatività e la fantasia nell'approccio al gioco.

Come l'AI ha rivoluzionato il mondo degli scacchi

Il gioco di AlphaZero ha evocato gli anni di gloria di Tal o di Kasparov, quando ogni mossa era eseguita per puntare direttamente al re avversario e i sacrifici strategici

[5] Per visionare l'analisi del motore, visitare il sito lichess.org/analysis.

erano all'ordine del giorno.

Le sue partite hanno aperto nuove prospettive nel mondo degli scacchi, scovando un nuovo stile di gioco in grado di superare le capacità dei motori tradizionali come Stockfish. Questo successo non deriva solo dalla potenza di calcolo bruta, ma anche da un approccio che sembra trarre vantaggio da strategie e intuizioni che vanno al di là delle capacità di calcolo puramente computazionale.

Da acerrimo avversario a compagno inseparabile

Siamo giunti al punto in cui, non potendo più competere con loro, è diventato essenziale allearsi con i computer. Grazie alla loro velocità di calcolo e alla capacità di esplorare milioni di posizioni in pochi secondi, i computer possono analizzare una partita in modo significativamente più dettagliato e accurato rispetto agli esseri umani, i quali potrebbero impiegare ore per eseguire un'analisi paragonabile.

Oggi, i motori di scacchi sono diventati strumenti essenziali per i giocatori di tutti i livelli nel loro percorso di miglioramento e di crescita. Uno dei principali utilizzi è l'analisi post-partita: dopo aver disputato un incontro di scacchi, un giocatore può caricare le proprie mosse su un motore e ottenere una valutazione istantanea della propria partita. Questa analisi permette ai giocatori di identificare gli errori commessi e le mosse migliori mancate, aiutandolo a non sbagliare più in futuro. Inoltre, sono strumenti preziosi per lo studio e la preparazione delle aperture, per testare nuove idee e analizzare posizioni critiche.

Tuttavia, è importante ricordare che, nonostante i grandi progressi tecnologici, il gioco degli scacchi rimane essenzialmente un'attività umana. I giocatori, indipendentemente dal loro livello, non dovrebbero semplicemente adottare le decisioni di un motore come definitive. Capire il mero significato di una mossa può richiedere una profondità di analisi che non è immediatamente evidente né facilmente riconducibile a un piano strategico chiaro: una mossa suggerita da un motore, che può sembrare non intuitiva, potrebbe nascondere una logica profonda che emerge solo attraverso varianti complesse estese oltre le 15/20 mosse. Il cervello umano, in casi eccezionali, non dispone della capacità di calcolare numeri di tale portata ... Una valutazione di +1.00 da parte di Stockfish indica un vantaggio significativo che, se mantenuto fino alla fine, può condurre alla vittoria. Questo vantaggio è gestibile per un motore, ma gli umani non sempre riescono a eseguire le mosse precise necessarie per sfruttarlo pienamente. Serve una grande dose di accuratezza e precisione: su una media di 20 o 30 mosse disponibili, spesso solo una è quella corretta per capitalizzare il vantaggio.

Il *fair play* nelle competizioni ufficiali

Sebbene siano evidenti i vantaggi dei software di scacchi, si devono considerare anche gli utilizzi negativi, come il *cheating informatico*, ovvero l'uso non autorizzato di motori di scacchi durante tornei ufficiali. Questa pratica è illegale e compromette sia la classifica del torneo che le vere capacità del giocatore.

Fortunatamente, le piattaforme online hanno adottato misure di sicurezza avanzate, come algoritmi anti-frode e monitoraggio delle mosse anomale, che sono in grado di beccare in modo istantaneo gli utenti che provano a fare i furbi.

Considerazioni finali

L'avvento dei motori scacchistici ha rivoluzionato il mondo degli scacchi, offrendo sia nuove opportunità che sfide. Questi strumenti potenti hanno reso accessibili analisi e strategie avanzate a giocatori di ogni livello, democratizzando la conoscenza scacchistica e permettendo a dilettanti e professionisti di migliorare le proprie abilità in modo significativo.

L'interazione tra intelligenza artificiale e talento umano ha elevato il gioco a livelli senza precedenti, spingendo i limiti della comprensione e della creatività. Un gioco così antico e prestigioso come gli scacchi ha sfruttato le tecnologie moderne per raggiungere vette mai toccate prima. Questa evoluzione ha spesso messo in discussione il gioco e lo stile dei grandi campioni del passato e, da Garry Kasparov in poi, anche dei migliori giocatori contemporanei.

La domanda da porsi è se, in futuro, vedremo intelligenze artificiali ancora più potenti e strabilianti degli attuali giganti presenti sul panorama mondiale: se in meno di cent'anni i risultati sono stati questi, chissà cosa potremmo aspettarci nei secoli a venire. Un altro interrogativo è se il gioco degli scacchi diventerà mai un *gioco risolto*,[6] nel quale sarebbe sufficiente che entrambi i giocatori, per non perdere, eseguano un algoritmo di mosse ben definito. Un ulteriore aspetto da considerare è l'impatto di eventuali impianti di mini computer nel cervello, dotati di un database di aperture e di finali di partita: come cambierebbero le regole del gioco? Gli scacchi resterebbero un'attività attraente e stimolante come lo sono stati fino ad oggi, o le nuove AI finiranno per far perdere interesse per i pezzi bianchi e neri? Qualunque sia il futuro, sarà fondamentale trovare un equilibrio tra tecnologia e creatività umana, per mantenere viva la magia degli scacchi.

[6] In un gioco risolto, ogni mossa sarebbe calcolata al massimo livello di precisione, portando a un esito inevitabile di vittoria, sconfitta o patta. Con una strategia ottimale, il risultato è già predeterminato e non dipende dagli errori dei giocatori. Esempi di giochi risolti sono il tris, la dama, il nim.

Ringraziamenti

Desidero esprimere la mia più profonda gratitudine a tutte le persone che mi hanno accompagnato e sostenuto nel mio percorso di studio e di crescita personale durante questi anni.

In primo luogo, un sentito ringraziamento va al mio relatore, il prof. Italo Epicoco, per la disponibilità, i consigli e il supporto dimostratomi durante la stesura di questa tesi. Un ringraziamento altrettanto doveroso va al mio correlatore, prof. Massimo Cafaro, per avermi guidato nella scelta dell'argomento e nella proposta del modello di machine learning per lo sviluppo del motore.

Sono grato anche a tutti i miei professori che hanno contribuito alla mia formazione, sia professionale che personale. Alcuni esami hanno richiesto un impegno notevole, talvolta più del previsto, ma ogni sfida affrontata ha rappresentato per me un'opportunità di crescita, preparandomi al meglio per il futuro che mi attende.

Un ringraziamento profondo va alla mia famiglia, in particolare ai miei genitori, per il loro amore incondizionato e per aver sempre creduto in me, anche nei momenti più difficili, e per avermi spinto a dare sempre il massimo in ogni situazione. Senza di voi non sarei arrivato fin qui. Un pensiero affettuoso va ai miei fratelli, Angelica e Michele: vi voglio un mondo di bene e mi sento onorato a condividere con voi non solo il legame di sangue, ma anche molto di più.

Un pensiero particolare va ai miei due "angeli custodi", il nonno Gino e la nonna Ada. Nonno, quante serate abbiamo trascorso insieme: tu a seguire le partite di Serie A, e io accanto a te, con il Mac davanti, immerso nello studio. Quante volte a tifare insieme per il Lecce e a discutere di politica e attualità. E tu, nonna, nonostante sono anni che non mi parli più, i tuoi occhi lo fanno per te, e trasmettono tutto il tuo amore e la tua sincerità. Sono grato di aver potuto condividere tanti momenti con te prima che la malattia cambiasse tutto.

Non posso dimenticare i miei amici e i miei colleghi di corso, con i quali ho condiviso non solo lo studio, ma anche momenti di risate, confronto e difficoltà. Sono sicuro che questi legami andranno ben oltre la fine del nostro percorso accademico.

Infine, un ringraziamento speciale va al Circolo Scacchi Alekhine di Aradeo, che

fin dalla mia tenera età ha alimentato la mia passione per il gioco degli scacchi. Una menzione in particolare va al presidente Antonio Longo e agli istruttori Luigi De Rosa, Orlando Pino e Mattia Renna: grazie per le serate trascorse a quadrigliare e per avermi svelato i segreti di questo meraviglioso gioco.

Appendice A

Le mosse dei pezzi

I pezzi degli scacchi si muovono su un tavolo da gioco, chiamato *scacchiera*, composto da 64 caselle disposte in una griglia 8×8. Metà delle 64 caselle sono di colore scuro (case camposcure), mentre l'altra metà sono di colore chiaro (case campochiare). All'inizio di una partita, i pezzi presenti sul campo di battaglia occupano esattamente metà delle caselle totali disponibili: ogni giocatore dispone di 16 pezzi: un re, una regina, due torri, due cavalli, due alfieri e otto pedoni. A turno, ciascuno dei due giocatori muove uno dei propri pezzi. Ogni pezzo ha un movimento distintivo, che lo rende unico e gli conferisce un ruolo specifico nella strategia del gioco.

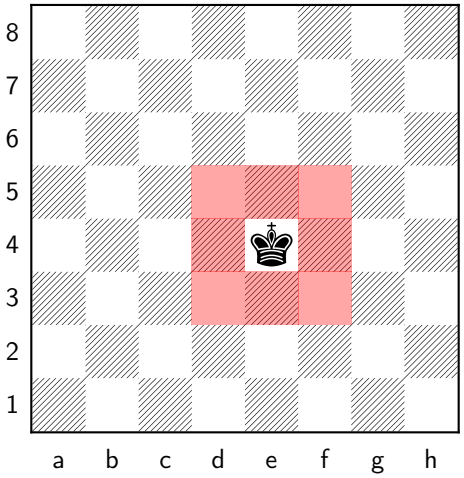
Il Re ♔

Il re è il pezzo più importante, tant'è che è l'unico pezzo che rimane sempre sulla scacchiera, anche quando subisce uno scacco matto. Nonostante ciò, sua maestà non è il pezzo più potente in termini di movimento. Si sposta di una casella in qualsiasi direzione: orizzontale, verticale o diagonale.

Quando il re è minacciato da un pezzo avversario, si dice che è *sotto scacco* e deve necessariamente sottrarsi alla minaccia. Questo può avvenire in tre modi:

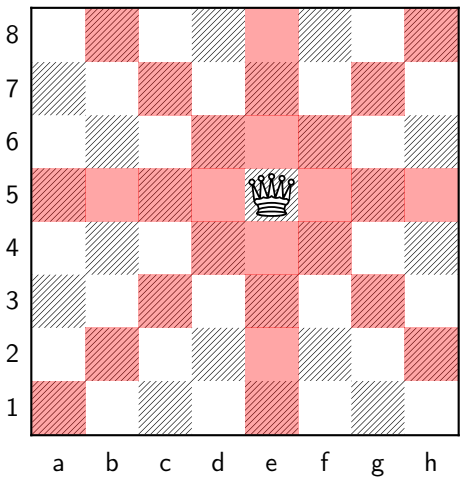
- spostandosi in una casella non controllata da alcun pezzo avversario,
- interponendo un proprio pezzo tra il re e il pezzo avversario che minaccia lo scacco,
- catturando il pezzo avversario che lo sta attaccando.

Se il re non può sottrarsi alla minaccia di scacco, la partita termina e si annuncia lo scacco matto. È importante notare che il re non può mai muoversi in una casella minacciata da un pezzo avversario, ossia non può mettersi volontariamente sotto scacco.



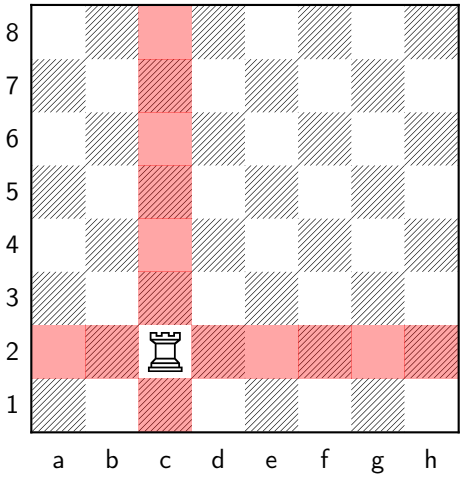
La Regina ♕

La regina (chiamata anche donna) si muove di un numero qualsiasi di caselle in orizzontale, verticale o diagonale. È il pezzo con maggiore possibilità di movimento del gioco. Quando si trova al centro della scacchiera, può controllare un massimo di ben 27 caselle (su un totale di 64, il 42%).



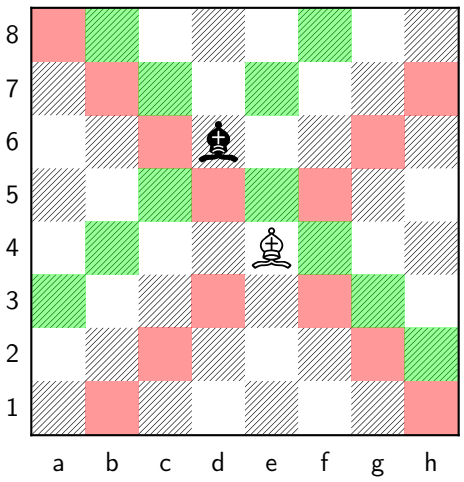
La Torre ♖

La torre si muove di un numero qualsiasi di caselle in orizzontale o verticale.



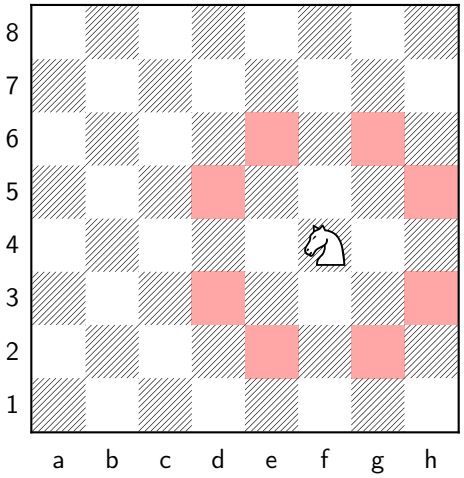
L'Alfiere ♗

L'alfiere si muove di un numero qualsiasi di caselle in diagonale. Ogni alfiere rimane sempre sulle caselle del colore iniziale per tutta la partita, distinguendosi quindi in alfiere campochiaro e alfiere camposcuro. Mantenere la coppia di alfieri durante la partita può offrire un grande vantaggio, soprattutto quando i pezzi iniziano a diminuire, poiché insieme possono controllare efficacemente l'intera scacchiera.



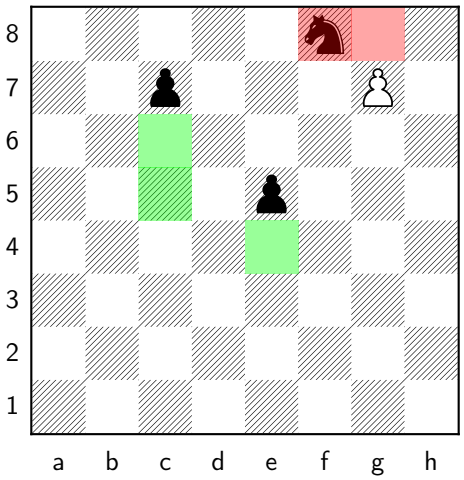
Il Cavallo ♘

Il cavallo si muove ricordando molto una forma ad "L": una casella in una direzione (orizzontale o verticale) e poi una casella in diagonale in avanti. È l'unico pezzo che può saltare sopra altri pezzi, siano essi propri o dell'avversario. Il cavallo cattura solo i pezzi avversari che si trovano nella casella di destinazione.



Il Pedone ♙

Il pedone si muove in avanti di una casella alla volta, ma cattura in diagonale (sempre di una casella). Alla sua prima mossa, può muoversi in avanti di due caselle. Se un pedone raggiunge l'ottava traversa (la traversa 8 per il Bianco, la 1 per il Nero), viene promosso a un altro pezzo a piacere, tranne il re (di solito una regina).



Eccezioni e mosse speciali

Arrocco

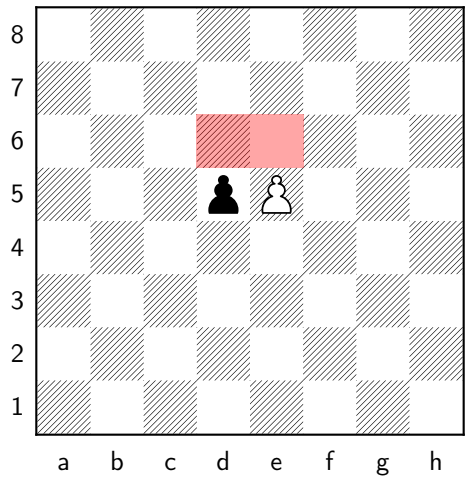
Una mossa particolare del re e della torre che permette al re di muoversi di due caselle verso una torre, e poi la torre si muove oltre il re. Ci sono due tipi di arrocco: corto (se l'arrocco avviene sul lato di re) e lungo (se avviene sul lato di donna). Le condizioni per l'arrocco sono:

- il re e la torre coinvolta non devono essersi mossi precedentemente,

- non ci devono essere pezzi tra il re e la torre,
- il re non deve essere sotto scacco, né deve passare per una casella controllata da un pezzo avversario.

En passant

Una mossa speciale di cattura che può avvenire quando un pedone muove di due caselle dalla sua posizione iniziale e finisce accanto a un pedone avversario. L'avversario può catturare quel pedone *en passant*, come se si fosse mosso di una sola casella. Tale mossa deve essere eseguita subito, altrimenti perde la sua validità.



Promozione

Come accennato poco fa, non appena un pedone giunge alla fine della scacchiera, viene sostituito da un qualunque pezzo tra regina, torre, alfiere e cavallo, indipendentemente dai propri pezzi che sono stati catturati in precedenza.

Appendice B

Notazione degli Scacchi

La notazione degli scacchi è un sistema di scrittura utilizzato per registrare le mosse effettuate durante una partita. Ci sono vari tipi di notazioni: la notazione algebrica e la notazione descrittiva.

Nella notazione algebrica, la più comune, le colonne della scacchiera sono contrassegnate con le lettere dalla a alla h, mentre le righe sono contrassegnate con i numeri da 1 a 8. Ogni casa della scacchiera è quindi identificata da una coppia di una lettera e un numero, come $e4$ o $d5$.

Le mosse sono scritte usando l'iniziale del pezzo mosso, che può variare a seconda della convenzione:

- iniziali inglesi: R (rook), N (knight), B (bishop), Q (queen), K (king).
- iniziali italiane: T (torre), C (cavallo), A (alfiere), D (donna), R (re).
- simboli grafici: ♖(torre), ♘(cavallo), ♗(alfiere), ♕(donna), ♔(re).

I pedoni sono gli unici pezzi che non presentano una iniziale. Ad esempio, *Nf3* significa che un cavallo è stato spostato nella casa *f3*; oppure *d4* indica che un pedone è stato spostato nella casa *d4*.

Se una mossa comporta una cattura, viene aggiunta una "x" tra l'iniziale del pezzo e la casa di destinazione. Per esempio, *Nxf3* significa che il cavallo ha catturato un pezzo nella casa $f3$.

Per indicare una promozione, si aggiunge un "=" seguito dal pezzo promosso alla fine della mossa del pedone. Ad esempio, *e8=Q* significa che un pedone ha raggiunto l'ottava traversa e si è promosso a donna.

I segni "+" e "#" sono usati per indicare lo scacco e lo scacco matto, rispettivamente. Ad esempio, *Qf7+* indica che la donna si è spostata in $f7$ e ha dato scacco, mentre *Qf7#* significa che ha dato scacco matto.

Elenco delle figure

2.1 La radice comune dell'albero scacchistico 5
2.2 La scacchiera e la disposizione dei pezzi all'inizio di una partita di scacchi. 6
2.3 Claude Shannon durante il 3° Campionato del Mondo di Scacchi per Computer, Festival Ars Electronica, 1980. Fonte: www.flickr.com. . . 13
2.4 Posizione sulla scacchiera alla 17^a mossa della partita tra Louis Paulsen e Paul Morphy all'American Chess Congress (1857). 16
2.5 Paulsen - Moprhy; posizione sulla scacchiera dopo il sacrificio di donna di Morphy e la conseguente ricattura della regina da parte di Paulsen con il proprio pedone nella casella f3. 17
2.6 Paulsen - Moprhy, posizione sulla scacchiera alla 28^a mossa. Il Bianco ha già ceduto la Regina, e ora è costretto a sacrificare la torre in f1 per poter difendere la minaccia di matto. 18
2.7 Esempio di applicazione della Strategia A su un albero decisionale. . . 19
2.8 Continuazione dell'esempio precedente: le valutazioni delle posizioni nei nodi terminali si sono propagate ai nodi genitori seguendo i metodi illustrati. 20
2.9 Esempio di potatura alfa-beta applicata ad un albero decisionale scacchistico. 24
2.10 Esempi di tabelle di codifica 15×15 colorate per donna, torre, alfiere, cavallo e re. La velocità del movimento del pezzo diminuisce all'aumentare dell'intensità del colore rosso. L'alfiere presenta caselle bianche (segnate con una x) in quanto può occupare solo metà scacchiera. 31
2.11 Garry Kasparov impegnato nella sfida contro Deep Blue nel 1997. Autore: Adam Nadel. Fonte: AP1997 © 34
2.12 IBM Deep Blue - G. Kasparov, posizione sulla scacchiera alla mossa 36, 2° turno, 1997. 35

3.1 Interfaccia grafica del motore Fritz 12. 39
3.2 Logo di Stockfish. Fonte: stockfishchess.org. 40
3.3 La funzione gradino e la funzione sigmoidale a confronto. 49
3.4 AlphaZero vs Stockfish, mossa 33. Immortal Zugzwang Game. Dicembre 2017. 51
3.5 AlphaZero vs Stockfish, mossa 45. Immortal Zugzwang Game. Dicembre 2017. 51
3.6 AlphaZero vs Stockfish, mossa 51. Immortal Zugzwang Game. Dicembre 2017. 52

4.1 Indici delle caselle della scacchiera. 58
4.2 Esempio di dataset utilizzato per l'addestramento di ChessMind Pro, estratto da un foglio di calcolo Excel. Ogni riga rappresenta una singola disposizione della scacchiera, con i valori numerici che indicano i pezzi, la mossa da eseguire e altre informazioni utili. 60
4.3 Esempio di output durante l'addestramento di ChessMind Pro, il quale mostra informazioni utili sul progresso del training. 61
4.4 Posizione di scacchi, un po' utopistica, che evidenzia una piccola imprecisione nel calcolo computazionale di un motore scacchistico. . . 68

Elenco delle tabelle

2.1 Numeri di partite possibili dopo le prime mosse di gioco. 6

3.1 Numero di posizioni possibili da analizzare in un finale di scacchi con un determinato numero di pezzi. 38

4.1 Architettura del modello sequenziale MLP adottato, con dettagli sui layer, il numero di neuroni e i parametri da ottimizzare per ciascuno strato. 63

Bibliografia

[1] Andrea Manzo, Augusto Caruso. *The Computer Chess World - How to make the most of chess software*. Aplha Chess - AI chess revolution, 2021.

[2] Michele Godena, Bruno Codenotti. *L'eterna sfida - Scacchi e Intelligenza Artificiale, da Turing ad Alphazero*. Hoepli, 2021.

[3] Matthew Sadler, Natasha Regan. *Game Changer - AplhaZero's Groundbreaking, Chess Strategies and the Promise of AI*. New in Chess, 2019.

[4] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow*. O'Reilly, 2019.

Webgrafia

1. www.chess.com
2. www.lichess.con
3. www.wikipedia.org
4. www.chessgames.com
5. www.chessprogramming.org