PokeBrain: Un'IA per Pokémon Showdown

Trotta Francesco Giuseppe
Ballacchino Giuseppe
Carlino Antonio

2 marzo 2025

Sommario

L'obiettivo del progetto è la creazione di un agente intelligente per Pokémon Showdown online, un simulatore di battaglie Pokémon, in grado di competere con avversari o altre intelligenze artificiali.

1 Introduzione

Il progetto mira a sviluppare un'IA in grado di competere su *Pokémon Showdown*, un simulatore di battaglie Pokémon online. L'IA sarà in grado di affrontare avversari umani o IA in tempo reale.

2 Pokemon

Pokémon è un franchise di giochi sviluppato da Game Freak e pubblicato da Nintendo nel 1996. Da allora ha generato una serie di sequel e spin-off che abbracciano diversi generi. La serie principale è composta da giochi di ruolo in cui il protagonista viaggia attraverso una regione, catturando creature chiamate Pokémon e addestrandole per usarle in battaglia. In questa sede discuteremo principalmente dell'aspetto delle battaglie nei giochi principali. Al momento della scrittura, ci sono state nove generazioni distinte di giochi Pokémon, ognuna delle quali ha introdotto cambiamenti significativi, pur mantenendo invariata la formula di base. In una battaglia, due squadre composte da un massimo di sei Pokémon si affrontano fino a quando tutti i Pokémon di una delle due squadre non sono resi inabili. Ogni generazione ha implementato nuovi sistemi e complessità nel sistema di combattimento:

- La Generazione 2 ha introdotto due nuove statistiche per il calcolo dei danni.
- La Generazione 3 ha assegnato a ogni Pokémon un'abilità passiva innata.
- La Generazione 4 ha ridefinito le statistiche utilizzate nel calcolo dei danni delle mosse.

E così via. Ognuna di queste generazioni ha avuto un impatto sostanziale sulla strategia competitiva. Poiché i nuovi giochi Pokémon vengono rilasciati regolarmente, il panorama strategico è molto fluido. Nonostante i numerosi cambiamenti dal 1996, le condizioni di vittoria sono rimaste invariate: l'obiettivo del combattimento è sconfiggere tutti i Pokémon dell'avversario. Prima dell'inizio di una battaglia, entrambi i giocatori costruiscono o ricevono squadre di sei Pokemon, ognuna delle quali ha fino a quattro mosse. Tutti i Pokemon hanno un valore noto come punti ferita (hp), che si esaurisce man mano che un Pokemon subisce danni dagli attacchi. Quando questo valore raggiunge lo 0, il Pokemon è considerato incapace e il gioco termina quando tutti i Pokemon di un giocatore raggiungono questo stato. Una volta che entrambi i giocatori hanno scelto una mossa, il gioco utilizza un sistema di priorità per risolvere gli eventi del turno. Tranne in casi molto specifici, il passaggio ha la precedenza sugli spostamenti, quindi vengono elaborate le mosse con priorità speciale. Se entrambi i giocatori hanno scelto mosse senza priorità speciale, il Pokémon con la statistica di Velocità più alta si muove per primo e l'altro Pokémon si muove per secondo. Vale la pena notare che i Pokemon possono essere inabilitati prima che si verifichi la loro mossa, nel qual caso il turno del giocatore viene effettivamente saltato.

3 Pokemon Showdown

Pokémon Showdown è un simulatore di battaglie Pokémon online, in cui è possibile creare squadre e competere contro altri giocatori. Abbiamo deciso di utilizzare Pokémon Showdown perché volevamo concentrarci esclusivamente sull'aspetto del combattimento, senza doverci preoccupare di navigare nella mappa, risolvere enigmi o catturare Pokémon, come avviene nei giochi classici. Inoltre, eravamo curiosi di sperimentare il formato delle battaglie casuali, in cui a ogni giocatore viene assegnata una squadra di sei Pokémon scelti casualmente. Questo approccio permette di focalizzarsi unicamente sulle decisioni strategiche da prendere durante la battaglia, senza dover considerare aspetti come la costruzione della squadra.

4 Showdown Framework

Utilizzeremo una versione modificata di Pokémon Showdown, un simulatore di lotta Pokémon open-source scritto in Node.js e sviluppato da Guangcong Luo. Questo framework opera preservando le meccaniche originali di Pokémon Showdown e fornendo un'interfaccia che emula la comunicazione basata su socket del server di Pokémon Showdown. Il framework, nel suo complesso, è suddiviso in quattro livelli principali:

- Livello di gioco: ospita lo stato di gioco canonico condiviso.
- Livello di comunicazione: facilita la comunicazione tra il motore di gioco e gli altri oggetti.
- Livello di interfaccia: analizza le informazioni provenienti dal livello di comunicazione e aggiorna lo stato personale del gioco.
- Livello dell'agente: prende le decisioni da inviare allo stato di gioco canonico per la gestione del gioco.

Ogni giocatore ha uno stato di gioco personale, che replica il più possibile lo stato di gioco canonico basandosi esclusivamente sulle informazioni accessibili a quel giocatore. Questo stato personale differisce dallo stato di gioco canonico, poiché non può includere informazioni nascoste. Queste informazioni vengono quindi trattate come valori predefiniti nulli fino a quando non sono rivelate al giocatore. L'agente interagirà principalmente con oggetti delle seguenti classi:

- Battle: rappresenta uno stato del gioco, contenente informazioni sugli effetti sul campo e riferimenti agli oggetti della classe BattleSide.
- BattleSide: rappresenta un giocatore, includendo informazioni sugli effetti collaterali e riferimenti agli oggetti della classe BattlePokemon.
- BattlePokemon: rappresenta un singolo Pokémon, contenente informazioni sullo stato attuale del Pokémon.

Il codice per ciascuna di queste classi è presente nel file Battle-engine.js.

5 Reference API

Il framework Showdown fornisce agli agenti un set di funzioni per recuperare dati su vari aspetti del gioco. Queste funzioni di riferimento sono contenute nel file tools.js e sono accessibili tramite l'oggetto globale Tools. Sono disponibili le seguenti funzioni:

- Tools.getMove(moveid): Restituisce informazioni su una mossa identificata da moveid. Il formato dei dati restituiti per una mossa è definito in moves.js. I campi pertinenti nell'oggetto restituito includono: move.basePower, move.type, e move.status.
- Tools.getTemplate(species): Restituisce informazioni su una specie di Pokémon. Il formato del modello restituito è definito in pokemon.js. I campi pertinenti nell'oggetto restituito includono: template.baseStats, template.types e template.abilities.

- Tools.getEffectiveness(source, target): Restituisce un valore intero che rappresenta l'efficacia di un attacco proveniente dalla sorgente source contro un obiettivo target. Un valore pari a 1 indica superefficacia, -1 indica resistenza e 0 indica neutralità. Questa funzione non considera le immunità.

- Tools.getImmunity(source, target): Restituisce un valore booleano che rappresenta l'immunità del bersaglio target a un attacco proveniente dalla sorgente source. Il valore false indica che il bersaglio è immune.

I dettagli relativi alla sintassi e agli argomenti necessari per l'invocazione di queste funzioni possono essere trovati nella documentazione fornita con il framework.

6 Forward Model

La classe Battle fornisce funzionalità per stimare i danni delle mosse e simulare turni completi nel gioco. Funzione getDamage(user, target, move)

Questa funzione stima i danni che la mossa move causerebbe se utilizzata da user su target. È importante notare che questa funzione tiene conto dello stato attuale del gioco per eseguire i suoi calcoli. Pertanto, lo stato di gioco utilizzato al momento della chiamata può influire sul risultato. Inoltre, poiché il calcolo dei danni in Pokémon include un certo grado di casualità (stochasticity), e dato che getDamage utilizza direttamente la formula del gioco per i danni, il risultato di questi calcoli è una stima e non una previsione esatta. Metodo choose(player, choice) Questo metodo consente agli agenti di simulare turni completi. Poiché i turni in Pokémon vengono elaborati in modo atomico, la simulazione di un turno può iniziare solo quando entrambi i giocatori hanno preso una decisione. Per questo motivo, choose deve essere chiamato due volte (una volta per ciascun giocatore) per far avanzare il modello. Formato del parametro choice Il parametro choice è una stringa che può essere formattata nei seguenti modi:

- "move X": Indica la decisione di far usare al Pokémon attivo la mossa X.

- "switch X": Indica la decisione di sostituire il Pokémon attivo con quello posizionato nella squadra al posto X.

- "forceskip": Indica la decisione di saltare il turno corrente.

Una volta che entrambi i giocatori hanno preso una decisione, lo stato di gioco avanza automaticamente, simulando tutti gli eventi che si verificano fino alla prossima richiesta di una mossa. È importante notare che, a causa della casualità e delle informazioni nascoste presenti nel gioco, questa simulazione rappresenta una stima dello stato risultante del gioco.

7 Sfida e Valutazioni

Pokemon può essere utilmente confrontato e contrapposto ad altri giochi dal punto di vista delle caratteristiche che influenzano il gameplay dell'IA come segue:

7.1 Branching Factor

Gli agenti di ricerca negli alberi saranno interessati al numero di risultati che potrebbero emergere da un singolo stato. Ignorando le ulteriori complicazioni legate alla stocasticità, ciò si riduce principalmente al numero di scelte che un giocatore può effettuare in un dato momento. In Pokémon, questo numero varia in base allo stato. In uno stato "pulito", privo di effetti o fattori modificanti, ci si può aspettare che il fattore di ramificazione sia compreso tra quattro e nove: quattro possibili mosse e cinque possibili cambi di Pokémon. In generale, questo è lo scenario più comune nel gioco e, come tale, una stima ragionevole del fattore di ramificazione medio è nove.

Effetti come **Encore** e **Fire Spine** possono ridurre questo fattore di ramificazione per il bersaglio, poiché tali effetti limitano l'accesso a determinate opzioni. Al contrario, alcune mosse richiedono scelte aggiuntive dopo essere state utilizzate. Ad esempio, **Baton Pass** obbliga l'utente a cambiare Pokémon dopo il suo utilizzo e, quindi, questa opzione può essere suddivisa in cinque sotto-opzioni, una per ogni

Pokémon disponibile in squadra. Pertanto, il fattore di ramificazione teorico massimo è 25: quattro mosse che portano a un cambio verso cinque Pokémon più cinque cambi diretti.

7.2 Numero di turni e Loop infiniti

Una delle prime considerazioni da fare quando si progetta un algoritmo per affrontare qualsiasi sfida è comprendere fino a che punto l'agente debba essere in grado di prevedere il futuro. Nel contesto del gioco umano, la maggior parte delle battaglie termina entro il turno 60, ma è importante sottolineare che non esiste un limite massimo teorico. Questo calcolo non tiene conto del ciclo volontario. Nel caso in cui entrambi i giocatori decidano di cambiare Pokémon a ogni turno, il gioco può teoricamente continuare all'infinito senza mai concludersi. Ciò può accadere se entrambi gli agenti riconoscono che cambiare Pokémon nella situazione attuale rappresenta la mossa ottimale, solo per scoprire che anche l'avversario ha cambiato Pokémon, rendendo quindi ottimale tornare al Pokémon originale. Nel gioco umano, queste situazioni si verificano raramente, ma gli agenti potrebbero non essere progettati per identificare e interrompere questi cicli. Poiché nessuno dei due giocatori sta intenzionalmente bloccando l'altro in un ciclo infinito, entrambi risultano "colpevoli" in questo scenario. Per prevenire tali situazioni, potrebbe essere introdotta una regola che stabilisca un limite massimo di turni, progettata specificamente per contrastare questa eventualità.

7.3 Elaborazione simultanea

Sebbene Pokémon sia classificato come un gioco a turni, gestisce i turni in modo leggermente diverso rispetto ad altri giochi dello stesso genere, come gli scacchi o Hearthstone. In Pokémon, invece di elaborare le decisioni dei giocatori in ordine alternato, le decisioni di entrambi i giocatori vengono elaborate simultaneamente.

Questo implica che lo stato del gioco può subire cambiamenti significativi tra il momento in cui un giocatore prende una decisione e il momento in cui tale decisione viene effettivamente eseguita. Ad esempio, all'inizio di un turno, il giocatore 1 decide di attaccare. Una volta che entrambi i giocatori hanno preso le loro decisioni, si scopre che il Pokémon del giocatore 2 possiede una statistica di Velocità superiore e quindi si muove per primo. L'attacco del giocatore 2 inabilita il Pokémon del giocatore 1, invalidando così la mossa pianificata da quest'ultimo.

Poiché situazioni di questo tipo possono verificarsi, gli agenti potrebbero prendere decisioni basate su stati del gioco già obsoleti, rendendo fondamentale che l'algoritmo sia in grado di gestire e adattarsi a tali circostanze dinamiche.

7.4 Mosse con effetti alternativi

Il confronto dei valori di salute rappresenta un metodo valido, ma limitato, per valutare la preferenza degli stati nei Pokémon. Questa metrica fornisce una visione parziale della situazione complessiva, poiché molte mosse del gioco non infliggono danni diretti, ma piuttosto producono effetti alternativi, come modificare le statistiche, infliggere stati alterati o applicare effetti di campo.

A differenza di valori come la salute o le statistiche, che possono essere misurati direttamente, gli stati alterati e gli effetti sul campo sono spesso di natura categorica. Ad esempio, è difficile quantificare con precisione il valore di un'ustione rispetto a una paralisi, rendendo complesso il confronto tra questi due stati.

Questa difficoltà è aggravata dal fatto che molte categorie di effetti includono un ampio numero di valori potenziali e non sono necessariamente mutuamente esclusive. Un singolo Pokémon può essere influenzato da più effetti contemporaneamente. Anche se ogni effetto può essere quantificato individualmente, rimane il problema di come combinare tali valori. Gli effetti possono essere correlati, interdipendenti o amplificarsi a vicenda, sollevando questioni sulla metodologia più adatta per gestire queste interazioni in modo coerente e significativo.

8 Stocasticità Pokemon

La progettazione di un agente per il gioco dei Pokémon deve tenere conto della stocasticità, un elemento cruciale che influenza profondamente la strategia e la valutazione delle opzioni disponibili.

In un sistema completamente deterministico, l'esito di ogni azione può essere previsto con esattezza. Tuttavia, nei Pokémon, molti elementi del gioco includono variabili casuali che rendono i risultati imprevedibili. Di conseguenza, le previsioni degli agenti si basano su stime probabilistiche piuttosto che su certezze.

Fonti di Stocasticità nei Pokémon La casualità nei Pokémon si manifesta su diversi livelli, influenzando profondamente il gameplay competitivo:

- *Danno Stocastico*: La formula del calcolo del danno include un moltiplicatore casuale che varia tra 0,85 e 1,0. Questa variazione significa che lo stesso attacco può infliggere quantità di danno diverse in situazioni simili.
- *Colpi Critici*: Ogni mossa d'attacco ha una probabilità innata del 6,25% di essere un colpo critico, che infligge il 50% in più di danno. Questo elemento aumenta l'incertezza durante la pianificazione delle strategie.
- *Precisione delle Mosse*: Ogni mossa ha un valore di precisione (compreso tra 0 e 100), che rappresenta la probabilità percentuale di successo.

  Mosse potenti come **DynamicPunch** o **Zap Cannon** hanno una bassa precisione, bilanciando il loro potenziale distruttivo. Gli effetti che alterano la precisione, come le condizioni meteorologiche, giocano un ruolo strategico significativo.
- *Effetti Secondari*: Molti attacchi hanno una probabilità di attivare effetti aggiuntivi. Questi effetti possono influire enormemente sul valore della mossa:

  Scald: 30% di probabilità di infliggere un'ustione.

  Air Slash: 30% di probabilità di far sussultare l'avversario, costringendolo a saltare il turno. Interazione con le Condizioni Meteorologiche Gli effetti atmosferici influenzano ulteriormente la stocasticità, alterando le probabilità di successo di determinate mosse:

  Pioggia: aumenta la precisione di Thunder dal 70% al 100%, rendendo la mossa più affidabile.

  Grandine: con l'abilità Mantello di Neve, diminuisce del 20% la precisione degli attacchi avversari.

9 Informazioni nascoste

Le informazioni nascoste esistono principalmente su due livelli nei Pokémon. A livello individuale, le mosse, le abilità e le statistiche di un Pokémon avversario sono nascoste. L'abilità di un Pokémon, sebbene abbia un grande impatto, può essere una di tre opzioni possibili e, per molti Pokémon, esiste un'unica scelta corretta da considerare. Questo problema può quindi essere in gran parte risolto applicando una conoscenza approfondita del gioco. Le statistiche di un Pokémon possono variare notevolmente in base a valori nascosti, ma l'applicazione di conoscenze relative alle build standard (o "cookie-cutter") può mitigare significativamente questo problema.

Le mosse, tuttavia, possono essere più difficili da prevedere. Sebbene un Pokémon possa entrare in battaglia con un massimo di quattro mosse, queste possono essere scelte da un learnset potenzialmente molto ampio. Ad esempio, Pikachu ha un learnset che include oltre 100 mosse. Anche in questo caso, la conoscenza del gioco risulta utile per eliminare mosse che sono in gran parte superate o non praticabili in un contesto competitivo. Tuttavia, anche dopo aver ristretto le possibilità, è improbabile che un agente sia in grado di prevedere con precisione tutte e quattro le mosse di un avversario.

È relativamente semplice simulare abbastanza avanti per sconfiggere l'attuale Pokémon avversario, ma senza alcuna informazione su quale potrebbe essere il prossimo Pokémon, diventa difficile formulare un piano che rimanga efficace per il resto della battaglia.

Per i formati in cui i giocatori costruiscono le proprie squadre, la conoscenza del gioco può essere applicata per identificare le sinergie tra Pokémon, restringendo così le possibilità. Tuttavia, nei formati in cui le squadre vengono generate casualmente, non esistono tali indizi da sfruttare. Questo rappresenta una sfida nei formati casuali, dove una previsione accurata oltre un certo punto è quasi impossibile a causa del vasto numero di possibilità in gioco.

10 Abilità "illusione"

C'è uno scenario molto specifico in cui la conoscenza del gioco da parte di un giocatore risulta semplicemente errata. Zoroark possiede un'abilità unica nota come Illusione, che gli consente di travestirsi da un altro Pokémon in battaglia. Questo crea un tipo speciale di informazioni nascoste persistenti, in quanto qualsiasi Pokémon utilizzato dall'avversario potrebbe in realtà essere uno Zoroark sotto mentite spoglie. Questo è particolarmente problematico perché il tipo Buio di Zoroark gli conferisce un'immunità agli attacchi di tipo Psichico.

Di conseguenza, un giocatore che presume erroneamente che un attacco di tipo Psichico sia efficace potrebbe tentare di utilizzarlo, fallendo. Se un agente non dispone di strumenti per riconoscere il fallimento o attribuirlo alla possibilità che l'avversario sia segretamente uno Zoroark, il giocatore potrebbe ripetere lo stesso attacco, ottenendo nuovamente un risultato inutile. Tuttavia, prendere decisioni tenendo costantemente in considerazione la possibilità che il Pokémon avversario sia camuffato introduce una grande incertezza nel processo decisionale strategico.

Al momento, Zoroark è l'unico Pokémon nei formati competitivi standard a possedere questa abilità, il che mitiga in parte il problema dell'attribuzione di fallimenti imprevisti. Detto ciò, a causa delle circostanze straordinarie e delle sfide presentate da questo singolo Pokémon, Zoroark è bandito dalla studio.

11 Conoscenza del dominio

Il Combattimento Pokémon include una serie di sottosistemi e considerazioni specifiche del dominio che i giocatori utilizzano nel processo decisionale strategico. Questi aspetti possono essere sfruttati anche dagli agenti per ottimizzare le decisioni, migliorare la strategia complessiva e accelerare il processo decisionale.Come detto precedentemente, la serie Pokemon è suddivisa in generazioni, e attualmente siamo alla nona generazione. Tuttavia, per semplificare lo sviluppo dell'IA, la nona generazione è stata esclusa dallo studio.

11.1 Caratteristiche dei Pokémon

Ogni Pokémon è definito da una serie di caratteristiche fondamentali che influenzano il suo comportamento in battaglia e il modo in cui può essere utilizzato strategicamente. Ecco una panoramica delle principali caratteristiche:

- **Natura:** Le nature influenzano la crescita delle statistiche, incrementando una specifica statistica del 10% e riducendone un'altra del 10%. Esistono 25 nature diverse.
- **Tipo:** Ogni Pokémon ha uno o due tipi (ad esempio, Fuoco, Acqua, Erba) che determinano:
  - **Resistenze e debolezze:** L'efficacia delle mosse contro altri tipi.
- **Statistiche Base:**
  - **HP:** Punti Salute, indicano la resistenza del Pokémon.
  - **Attacco:** Influisce sul danno inflitto dalle mosse fisiche.
  - **Difesa:** Riduce il danno ricevuto dalle mosse fisiche.
  - **Attacco Speciale:** Influisce sul danno inflitto dalle mosse speciali.
  - **Difesa Speciale:** Riduce il danno ricevuto dalle mosse speciali.
  - **Velocità:** Determina quale Pokémon si muove per primo in battaglia.
- **Mosse che aumentano le statistiche in battaglia:**
  - **Danza Spada:** Incrementa l'Attacco di due livelli.
  - **Calmamente:** Incrementa Attacco Speciale e Difesa Speciale.
- **Abilità:** Ogni Pokémon ha un'abilità unica o una tra più opzioni disponibili. Le abilità forniscono effetti passivi, ad esempio:

  - **Levitazione:** Rende immuni alle mosse di tipo Terra.
  - **Tossicità:** Infligge danno aggiuntivo se il Pokémon avversario è avvelenato.

- **Mosse:** La selezione di mosse con tipi diversi per massimizzare i danni contro vari avversari.
- **Effetti secondari:** Probabilità di infliggere stati alterati o modificare le statistiche.
  - **Paralisi:** Riduce la velocità e talvolta impedisce di attaccare.
  - **Ustione:** Riduce l'Attacco e infligge danni ogni turno.
  - **Congelamento e Sonnolenza:** Impediscono al Pokémon di agire per più turni.
- **Effetti di Campo:** Mosse o abilità che alterano il campo di battaglia:
  - **Pioggia:** Potenzia le mosse d'Acqua e indebolisce quelle di Fuoco.
  - **Campo Elettrico:** Potenzia le mosse di tipo Elettro e previene la sonnolenze.
- **IVs (Valori Individuali) e EVs (Valori di Allenamento):** Questi influenzano lo sviluppo delle statistiche e possono essere ottimizzati per creare Pokémon specializzati.
- **Dinamax e Mosse Z (nelle generazioni recenti):** Meccaniche che alterano drasticamente il gameplay aumentando temporaneamente il potere delle mosse o fornendo abilità straordinarie.
- **Affinità con l'Allenatore:** La felicità e la fedeltà possono influenzare il comportamento in battaglia, ad esempio aumentando i critici o diminuendo i danni ricevuti.

Queste specifiche aggiuntive rappresentano una parte essenziale per comprendere a fondo il potenziale strategico dei Pokémon.

11.2 Moltiplicatori e Tabella dei Moltiplicatori

La tabella dei tipi rappresenta uno degli aspetti più fondamentali del sistema di combattimento nei Pokémon. Ogni mossa del gioco è associata a un tipo, e tutti i Pokémon possiedono uno o due tipi. Quando una mossa viene utilizzata su un Pokémon, viene applicato un moltiplicatore al calcolo del danno, in base ai tipi coinvolti.

Se il tipo della mossa è *superefficace* contro il tipo del bersaglio, il danno inflitto viene raddoppiato ($2\times$). Se il bersaglio è *resistente*, il danno viene dimezzato ($0.5\times$). In caso di *immunità*, l'attacco non ha alcun effetto ($0\times$). Nel caso in cui il Pokémon bersaglio abbia due tipi, i moltiplicatori vengono calcolati separatamente per ciascun tipo e successivamente moltiplicati tra loro, il che può portare a moltiplicatori come $4\times$ (danno quadruplo) o $0.25\times$ (danno ridotto al 25%).

Esiste inoltre un moltiplicatore aggiuntivo di $1.5\times$, noto come *Same Type Attack Bonus* (STAB), che si applica se il tipo della mossa corrisponde al tipo del Pokémon che la utilizza.

Poiché i bonus e le penalità legati ai tipi hanno un grande impatto, le mosse supereefficaci infliggono generalmente molti più danni rispetto a quelle resistite.

A seguire la tabella dei moltiplicatori:

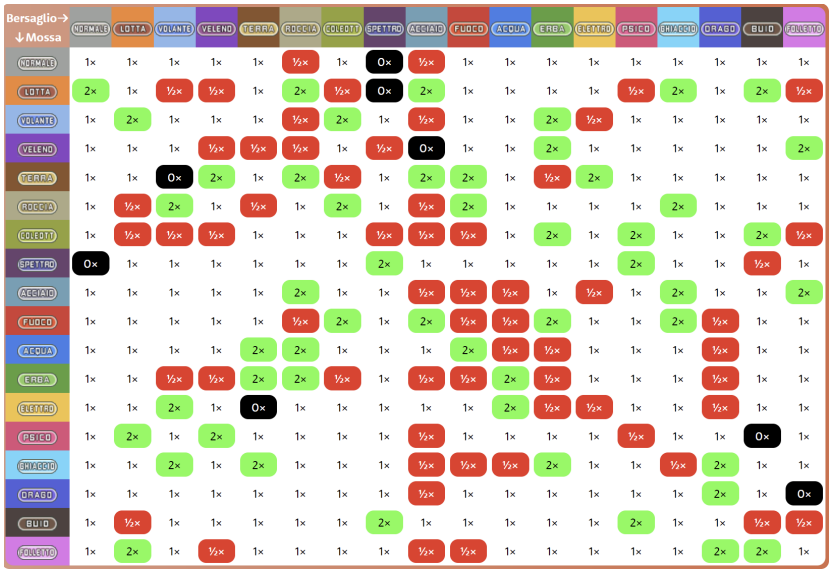

Figura 1: Tabella dei Moltiplicatori di Tipo

11.3 Archetipi

La gestione delle informazioni nascoste è un elemento cruciale. Ad esempio, ci sono diverse combinazioni di mosse ben note per la loro sinergia. Se si scopre che un Pokémon di tipo Erba conosce la mossa *Sunny Day*, è generalmente ragionevole presumere che abbia accesso anche alla mossa *Solarbeam*, poiché quest'ultima trae un beneficio diretto dall'effetto di *Sunny Day*.

Inoltre, molti Pokémon rientrano in determinati archetipi e sono progettati per utilizzare mosse che sfruttano al meglio le loro qualità. Ad esempio, è lecito ritenere che un Pokémon con un Attacco fisico estremamente elevato e un Attacco speciale mediocre userà quasi esclusivamente mosse fisiche per infliggere danni. Incorporando una conoscenza del dominio più specifica, è possibile prevedere le mosse in base agli archetipi a cui appartengono i singoli Pokémon.

Per esempio, esiste un gruppo di Pokémon nel gioco competitivo che rientra in un archetipo noto come *Spinners*. Questi Pokémon sono apprezzati soprattutto per il loro accesso alla mossa *Rapid Spin*. È quindi ragionevole presumere che gli *Spinners* abbiano *Rapid Spin* nel loro set di mosse.

A un livello più ampio, la conoscenza del dominio può essere utile per prevedere quali Pokémon possiede un avversario. È importante notare che tale utilizzo della conoscenza del dominio è applicabile solo nei formati in cui i giocatori costruiscono le loro squadre manualmente. Le squadre costruite manualmente sono spesso create attorno a una strategia centrale.

Riuscire a identificare la strategia centrale di un avversario permette di prevedere con una certa precisione la composizione della sua squadra. Per esempio, *Trick Room* è una mossa che inverte l'ordine con cui i Pokémon attaccano ogni turno. Se un giocatore utilizza questa mossa, è generalmente sicuro assumere che stia usando una squadra composta da Pokémon molto lenti ma estremamente potenti.

12 Mosse con ritardi

Diverse strategie in Pokémon prevedono l'uso di mosse che forniscono benefici molto tempo dopo che i Pokémon attuali sono stati sconfitti. Ad esempio, *Spikes*, *Toxic Spikes* e *Stealth Rock* sono mosse note come *entry hazard*, che danneggiano i Pokémon ogni volta che entrano in battaglia. Se usate correttamente, queste mosse possono causare fino al 75% dei danni alla salute massima di un Pokémon quando entra in battaglia. Sebbene queste mosse non infliggano danni immediati e non offrano vantaggi a breve termine, esse sono spesso considerate di basso valore dai giocatori che preferiscono guadagni rapidi.

13 Architettura dell'IA

L'idea è quella di implementare un algoritmo Minimax, ovvero un albero decisionale in cui ciascun giocatore, due in totale, dispone di quattro possibili mosse e cinque possibili cambi, per un totale di nove nodi per giocatore. Ogni nodo avrà una valutazione basata su un intervallo di possibili risultati.

Inizialmente, si conosce esclusivamente il primo Pokémon dell'avversario. Tuttavia, avendo la possibilità di gestire entrambi i giocatori, verrà creato un array contenente i sei Pokémon dell'avversario, in modo da ottenere un ambiente deterministico. Un oggetto Pokémon, dunque, includerà informazioni relative al tipo, alle statistiche e, soprattutto, alle mosse disponibili.

Man mano che il gioco procede, l'albero verrà esplorato a partire dalla radice, ossia dallo stato iniziale del gioco, per poi passare attraverso i nodi corrispondenti alle scelte del giocatore e successivamente a quelli dell'avversario. Seguendo la logica del Minimax, il giocatore sceglierà il nodo con il valore massimo, mentre l'avversario selezionerà il nodo con la valutazione minima.

La valutazione di ciascun nodo sarà determinata in base a diversi fattori, tra cui il danno subito dal proprio Pokémon, la sconfitta del proprio Pokémon, il danno inflitto all'avversario e la possibile eliminazione di un suo Pokémon. Inoltre, verrà assegnato un punteggio più elevato in base all'efficacia delle mosse, considerando le debolezze e le resistenze dei Pokémon in campo.

Prima dell'inizio della partita, verrà verificata la velocità relativa dei Pokémon in gioco, determinando se il Pokémon avversario sia più veloce rispetto al proprio. Questa informazione sarà fondamentale per stabilire l'ordine delle mosse e ottimizzare le strategie di combattimento.

13.1 Cos'è Minimax

L'architettura dell'agente intelligente basata sull'algoritmo Minimax per prendere decisioni in una battaglia su Pokémon Showdown può essere suddivisa in diversi moduli e componenti, che collaborano per generare decisioni ottimali in tempo reale.

Il Minimax è una tecnica che analizza le decisioni future costruendo un albero decisionale in cui il giocatore cerca di massimizzare il proprio vantaggio, mentre l'avversario tenta di minimizzarlo. L'algoritmo esplora tutte le possibili mosse disponibili, valutando le conseguenze di ogni scelta sulla base di una funzione di valutazione predefinita.

L'albero decisionale cresce a partire dallo stato attuale della battaglia, simulando tutte le possibili azioni dei Pokémon in campo, sia del giocatore che dell'avversario. Ad ogni livello dell'albero:

Il giocatore sceglie l'azione con il valore massimo per ottimizzare il proprio vantaggio. L'avversario sceglie l'azione con il valore minimo, cercando di ostacolare il giocatore. La valutazione dei nodi dipende da diversi fattori, come il danno inflitto, il danno subito, l'efficacia delle mosse, la velocità dei Pokémon e il numero di Pokémon ancora disponibili. Questo approccio permette di simulare il comportamento dell'avversario e scegliere la strategia ottimale in ogni turno, garantendo decisioni razionali e ben ponderate.

14 Struttura dei Dati e TeamManager

La gestione dei Pokémon avversari è affidata alla classe `TeamManager`, che permette di memorizzare e distinguere le squadre dei due giocatori, Player1 e Player2, attraverso un identificativo univoco (ID).

Le squadre vengono salvate in un dizionario, dove ogni chiave corrisponde all'ID di un giocatore e il valore è la sua squadra. La squadra dell'avversario viene estratta dall'oggetto `Battle` della libreria `PokeEnv` e salvata in un array contenente sei oggetti Pokémon.

Questo array viene poi passato a `PokeMinimax`, che lo utilizza per la costruzione dell'albero decisionale. L'albero inizia dalla radice, rappresentando lo stato iniziale della battaglia, e si sviluppa attraverso i nodi relativi alle azioni del giocatore e dell'avversario.

Ogni nodo dell'albero contiene:

- Il riferimento al nodo padre.
- Il team dell'avversario.
- Il Pokémon attivo dell'avversario.
- Un dizionario per tracciare gli HP dei Pokémon avversari.

- Il nostro Pokémon attivo.
- Un dizionario per tracciare gli HP dei nostri Pokémon.
- L'azione selezionata (mossa o cambio Pokémon).
- Una lista dei nodi figli.
- Il valore assegnato al nodo.

15 Scelta della Mossa e Switch di Pokémon

15.1 Metodo scegliMossa

Il metodo `scegliMossa` parte inizializzando i dizionari HP per entrambi i giocatori, in modo da tenere traccia degli HP attuali di ciascun Pokémon. Gli HP dell'avversario vengono estratti dall'array di Pokémon avversari salvato precedentemente.

Dopo aver creato il nodo radice dell'albero decisionale, il metodo verifica lo stato del nostro Pokémon attivo:

- Se il nostro Pokémon attivo è esausto (HP ≤ 0), si procede alla sostituzione con un altro Pokémon tramite il metodo `BestSwitch`.
- Se il Pokémon è ancora attivo, viene attivato il metodo `Minimax` per determinare la mossa ottimale.

15.2 Metodo BestSwitch

Il metodo `BestSwitch` gestisce la sostituzione dei Pokémon:

- Genera cinque nodi, uno per ciascun Pokémon disponibile in panchina.
- Ogni nodo viene valutato utilizzando `Minimax`, scegliendo il Pokémon con il punteggio più alto.
- Il miglior cambio viene selezionato e restituito come azione da eseguire.

15.3 Metodo Minimax

Il metodo `Minimax` esplora l'albero decisionale per determinare la mossa ottimale:

- Se viene raggiunto un nodo foglia o la battaglia è terminata (`IsTerminal`), il valore del nodo viene calcolato e restituito.
- Se è il turno dell'agente, viene selezionata la mossa con il valore più alto.
- Se è il turno dell'avversario, viene selezionata la mossa con il valore più basso, simulando la strategia avversaria.

Il metodo `IsTerminal` verifica se tutti i Pokémon di uno dei due giocatori sono esausti (HP ≤ 0), determinando la fine della partita.

16 Calcolo del Valore e Gestione dei Danni

Il calcolo del valore di ciascun nodo viene effettuato mediante il metodo `calcoloValore`, che analizza gli HP dell'avversario per valutare gli effetti delle mosse eseguite.

16.1 Generazione dei Figli e Modifica delle Statistiche

I metodi responsabili della generazione dei nodi figli modificano dinamicamente le statistiche e gli HP nei dizionari di riferimento. Si considerano tre scenari principali:

- L'avversario attacca per primo → Se il nostro Pokémon sopravvive, può contrattaccare.
- Il nostro Pokémon attacca per primo → Se il Pokémon avversario sopravvive, può contrattaccare.

16.2 Metodi di Supporto in Utils

Il modulo `Utils` fornisce metodi di supporto per la gestione della battaglia:

- Calcolo del danno → Determina il danno inflitto in base a statistiche e mosse.
- Aggiornamento degli HP → Tiene traccia dello stato di salute dei Pokémon.
- Calcolo della velocità → Determina chi attacca per primo.
- Valutazione dell'efficacia delle mosse → Determina se una mossa è super efficace o meno.

16.3 Valutazione del Nodo

Nel metodo `calcoloValore`, il valore del nodo è determinato dai seguenti fattori:

- Pokémon avversario sconfitto → Incremento della valutazione.
- Danno inflitto all'avversario → Moltiplicatore positivo (3x il danno subito dal Pokémon avversario).
- Mossa super efficace → Ulteriore bonus alla valutazione.
- Danno subito dall'agente → Penalizzazione.
- Pokémon dell'agente sconfitto → Significativa riduzione del valore.

17 Attivazione della Battaglia

Il metodo `ChooseMove` viene attivato ogni volta che il giocatore deve selezionare una mossa.

17.1 Gestione del Primo Turno

Nel primo turno, il team avversario non è ancora noto (`opponent_team` = null). In questo caso:

- Viene selezionata una mossa di default per garantire la continuità della partita.

Nei turni successivi:

- Il sistema analizza l'albero decisionale generato da `PokeMinimax` per scegliere la mossa ottimale.
- Se la mossa è null, viene selezionata una mossa di fallback.
- Se `PokeMinimax` non è stato inizializzato correttamente, il sistema notifica l'errore e utilizza una strategia predefinita.

18 Conclusioni e sviluppi futuri

L'intelligenza artificiale implementata per la gestione delle battaglie in Pokémon Showdown, sebbene capace di compiere scelte strategiche, presenta alcune limitazioni che ne influenzano l'efficacia complessiva. Sebbene l'IA sia in grado di selezionare mosse in modo abbastanza coerente, non sempre la scelta è ottimale. In particolare, si riscontrano delle difficoltà in alcune situazioni che portano a scelte subottimali, come la ripetizione di mosse identiche in sequenza, creando un ciclo che può manifestarsi più volte durante una partita. Questo comportamento, sebbene non infinito, può verificarsi con una sequenza di mosse ripetitive anche per 3-4 volte o più, senza una variazione che consenta all'IA di adattarsi alle circostanze.

Un altro problema che emerge riguarda l'inizializzazione della squadra avversaria: poiché l'IA non dispone del team completo dell'avversario fin dal primo turno, questo può portare a scelte di mosse non ottimali, in particolare nei casi in cui una mossa super efficace potrebbe portare alla sconfitta di un Pokémon già al primo turno. In questi casi, l'IA potrebbe non effettuare una mossa sicura, aumentando il rischio di una strategia fallimentare.

Inoltre, la valutazione dei nodi dell'albero decisionale potrebbe essere migliorata. Attualmente, la stima dei danni inflitti e subiti non tiene conto di variabili importanti, come i boost temporanei, gli oggetti equipaggiati dai Pokémon o altre modifiche temporanee delle statistiche. Un approccio più sofisticato alla valutazione del nodo, che includa questi fattori, potrebbe migliorare significativamente la precisione delle decisioni dell'IA. Un calcolo più approfondito dei danni, considerando tutti i parametri coinvolti, permetterebbe all'IA di prendere decisioni più accurate, riducendo il rischio di errori che potrebbero portare alla sconfitta.

Un'altra direzione interessante per migliorare l'IA sarebbe l'introduzione di tecniche di *machine learning*. Questi metodi permetterebbero all'intelligenza artificiale di apprendere dai propri errori e migliorare continuamente nel tempo, riducendo la probabilità di ripetere le stesse mosse sbagliate. L'uso del machine learning potrebbe anche contribuire a rendere l'IA più adattabile e meno prevedibile, migliorando la sua capacità di rispondere alle varie strategie degli avversari.

Nonostante queste problematiche, i risultati ottenuti finora sono comunque soddisfacenti. L'IA riesce a compiere scelte strategiche discrete e, in alcune situazioni, ad eseguire mosse super efficaci, dimostrando che il sistema è in grado di rispondere in modo competente a molte delle dinamiche di battaglia. Sebbene ci siano numerosi miglioramenti che potrebbero essere apportati per rendere l'IA più robusta e precisa, i progressi ottenuti fino ad ora sono promettenti, e con ulteriori ottimizzazioni, il comportamento dell'IA potrebbe diventare molto più raffinato e competitivo.