Report attività progettuale fondamenti di Intelligenza Artificiale M

pokemonDoubleBattlesIA

Autori:

Cristiano Arnaudo

Andrea Munari

Link al repository GitHub del progetto (con le istruzioni per l'esecuzione):
https://github.com/skyocrandive/pokemonDoubleBattlesIA

Introduzione

L'obiettivo di questa attività progettuale è quello di creare una intelligenza artificiale in grado di giocare alla modalità Lotte in doppio dei videogiochi di Pokémon.

Al fine di poter creare questa IA è stato scelto di fare uso di Pokemon Showdown: un simulatore open-source di lotte Pokémon che permette sia di creare il proprio team, sia di poter affrontare altre persone in vari formati. Oltretutto, è disponibile online la libreria Python poke-env che permette all'IA di interfacciarsi con Pokemon Showdown.

Fra i vari formati a disposizione è stato scelto il formato VGC 2021. Questo formato equivale al formato serie 7 dei videogiochi Pokemon Spada e Pokemon Scudo (generazione 8). È stato scelto questo formato in quanto i formati della generazione 5 in giù avevano un malfunzionamento nel team building e il gruppo aveva maggiore conoscenza della scena competitiva di ottava generazione. Inoltre, è stato deciso di non fare uso di meccaniche generazionali (dinamax, mosse z, mega evoluzioni per fare un esempio) per cercare di concentrarci sul gioco base privo di ulteriori complessità.

In questo progetto ci concentriamo sulla realizzazione di quattro giocatori:

1. **DoublesRandomPlayer**: non predicibile ma per nulla efficace in quanto sceglie a caso le proprie mosse.
2. **DoublesMaxDamagePlayer**: molto prevedibile, punta sempre a fare il massimo danno.
3. **DoublesTrueMaxDamagePlayer**: ha maggiore conoscenza delle complessità delle meccaniche di gioco rispetto al precedente (abilità, oggetti, terreni, clima). È in grado di calcolare con maggiore accuratezza i danni delle singole mosse.
4. **SmartPlayer**: giocatore capace di usare Protezione, cambiare Pokemon in caso di uno scontro sfavorevole, selezionare un'azione con un po' di varianza per rendere l'IA meno predicibile. Il giocatore è poco predicibile e agisce cercando di utilizzare tutte le conoscenze che può ottenere anche durante la battaglia stessa. Non è vincolato nell'uso di mosse d'attacco (a meno che non sia certo di poter sconfiggere l'avversario con una mossa di priorità).

Per testare ed eseguire queste IA fare riferimento al file README.md presente nel root del repository GitHub del progetto.

In questo report vedremo prima le IA prese singolarmente, poi un confronto tra i 4 giocatori realizzati e infine si conclude indicando i possibili sviluppi futuri del progetto.

DoublesRandomPlayer

È la prima IA che è stata realizzata con lo scopo di prendere confidenza con la libreria poke-env.

All'inizio della sfida sceglie in ordine casuale i Pokémon da utilizzare nella battaglia.

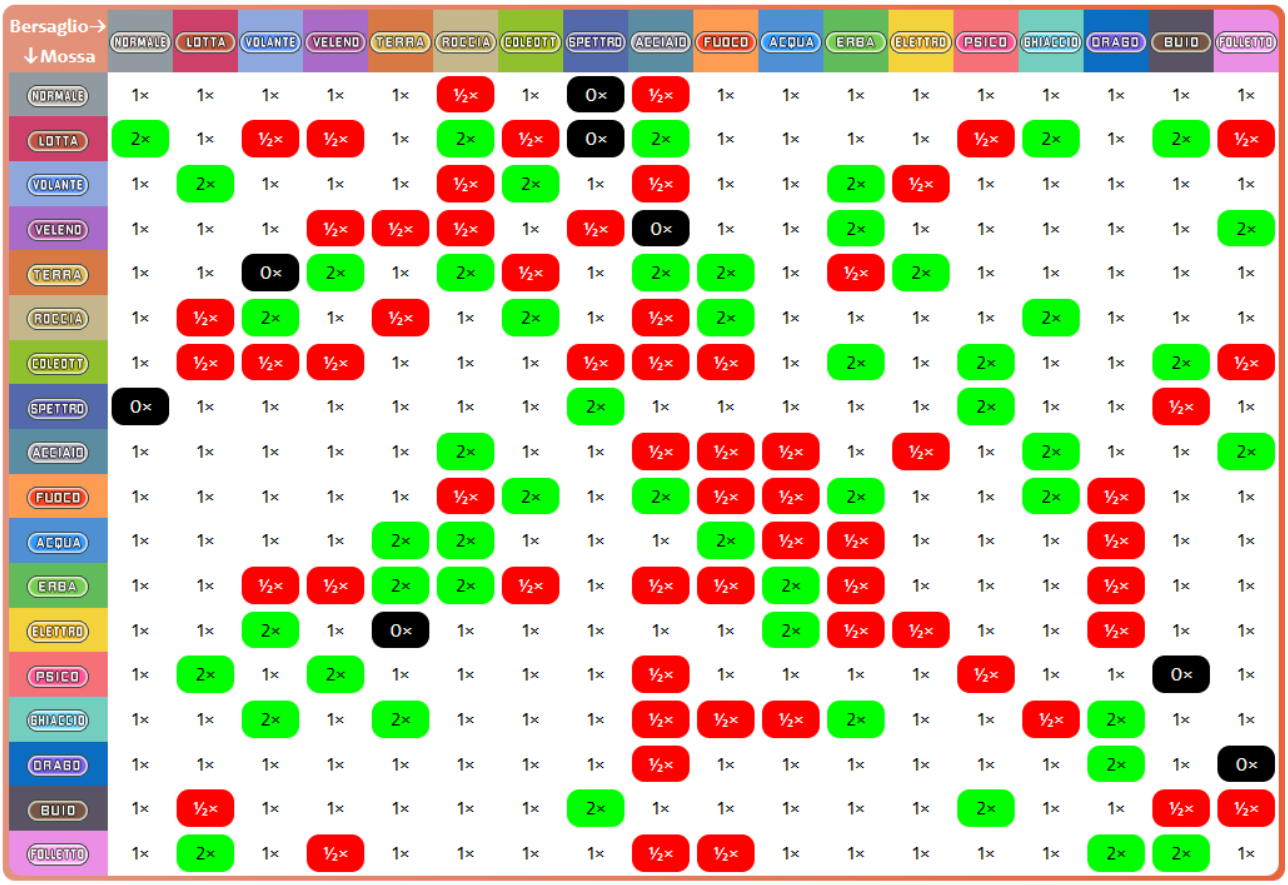
A ogni turno utilizza poke-env per estrarre:

- Tutte le possibili mosse a disposizione del Pokémon attivo
- Tutte le possibili sostituzioni di Pokémon che può eseguire il giocatore IA

Dopodiché elimina dalla lista ottenuta le mosse a target singolo che bersagliano un Pokémon alleato. Infine, sceglie casualmente l'azione da eseguire fra quelle rimaste.

DoublesMaxDamagePlayer

All'inizio della sfida i Pokémon da usare in battaglia vengono scelti in base al rapporto di resistenza fra i tipi del proprio Pokémon e i tipi dei Pokémon dell'avversario:



Bersaglio→ / ↓Mossa	NORMALE	LOTTA	VOLANTE	VELENO	TERRA	ROCCIA	COLEOTT	SPETTRO	ACCIAIO	FUOCO	ACQUA	ERBA	ELETTRO	PSICO	GHIACCIO	DRAGO	BUIO	FOLLETTO
NORMALE	1×	1×	1×	1×	1×	½×	1×	0×	½×	1×	1×	1×	1×	1×	1×	1×	1×	1×
LOTTA	2×	1×	½×	½×	1×	2×	½×	0×	2×	1×	1×	1×	1×	½×	2×	1×	2×	½×
VOLANTE	1×	2×	1×	1×	1×	½×	2×	1×	½×	1×	1×	2×	½×	1×	1×	1×	1×	1×
VELENO	1×	1×	1×	½×	½×	½×	1×	½×	0×	1×	1×	2×	1×	1×	1×	1×	1×	2×
TERRA	1×	1×	0×	2×	1×	2×	½×	1×	2×	2×	1×	½×	2×	1×	1×	1×	1×	1×
ROCCIA	1×	½×	2×	1×	½×	1×	2×	1×	½×	2×	1×	1×	1×	1×	2×	1×	1×	1×
COLEOTT	1×	½×	½×	½×	1×	1×	1×	½×	½×	½×	1×	2×	1×	2×	1×	1×	2×	½×
SPETTRO	0×	1×	1×	1×	1×	1×	1×	2×	1×	1×	1×	1×	1×	2×	1×	1×	½×	1×
ACCIAIO	1×	1×	1×	1×	1×	2×	1×	1×	½×	½×	½×	1×	½×	1×	2×	1×	1×	2×
FUOCO	1×	1×	1×	1×	1×	½×	2×	1×	2×	½×	½×	2×	1×	1×	2×	½×	1×	1×
ACQUA	1×	1×	1×	1×	2×	2×	1×	1×	1×	2×	½×	½×	1×	1×	1×	½×	1×	1×
ERBA	1×	1×	½×	½×	2×	2×	½×	1×	½×	½×	2×	½×	1×	1×	1×	½×	1×	1×
ELETTRO	1×	1×	2×	1×	0×	1×	1×	1×	1×	1×	2×	½×	½×	1×	1×	½×	1×	1×
PSICO	1×	2×	1×	2×	1×	1×	1×	1×	½×	1×	1×	1×	1×	½×	1×	1×	0×	1×
GHIACCIO	1×	1×	2×	1×	2×	1×	1×	1×	½×	½×	½×	2×	1×	1×	½×	2×	1×	1×
DRAGO	1×	1×	1×	1×	1×	1×	1×	1×	½×	1×	1×	1×	1×	1×	1×	2×	1×	0×
BUIO	1×	½×	1×	1×	1×	1×	1×	2×	1×	1×	1×	1×	1×	2×	1×	1×	½×	½×
FOLLETTO	1×	2×	1×	½×	1×	1×	1×	1×	½×	½×	1×	1×	1×	1×	1×	2×	2×	1×

Figura 1 - tabella rapporto efficacia/resistenza tipi Pokémon (https://wiki.pokemoncentral.it/Tipo)

A ogni turno calcola il danno che farebbe ogni mossa del Pokémon attivo contro gli avversari tenendo conto soltanto di:

- Potenza base della mossa
- Rapporto fra il tipo della mossa e il tipo del Pokémon che subirebbe la mossa
- Statistiche del Pokémon attivo e una predizione basilare delle statistiche del Pokémon che subirebbe la mossa
- Nel caso di mosse che colpiscono più bersagli il danno totale calcolato sarà la somma dei danni su entrambi i Pokémon avversari attivi moltiplicato per 0.75

Questi calcoli vengono fatti all'interno di una classe ausiliaria BattleUtilities.

Similmente al *DoublesRandomPlayer*, non sceglie come bersaglio delle proprie mosse gli alleati.

DoublesTrueMaxDamagePlayer

Ha lo stesso comportamento del *DoublesMaxDamagePlayer*, ma sostituisce la funzione del calcolo dei danni con una che tiene conto delle seguenti meccaniche di gioco:

- Clima (pioggia e sole)
- Campi (campo elettrico, erboso, psichico, nebbioso)
- EV (Effort Values - Valori di sforzo in taliano) e IV (Individual Values – Valori Individuali in italiano) se conosciuti, se non lo sono imposta IV a 31 e EV a 52 nelle statistiche offensive e difensive, in HP e velocità imposta IV a 31 e EV a 0
- Mosse con potenza base variabile come “acrobazia” (acrobatics) e “vortexpalla” (giro ball)
- La presenza di “schermo luce” (light screen) “riflesso” (reflect) e “velaurora” (aurora veil)
- Presenza di alterazioni di stato come bruciatura e paralisi
- Abilità come parafulmini (lightning rod) e dentistretti (guts)
- Oggetti come bandascelta (choice band) e lentiliscelta (choice specs)
- Aumenti/cali di statistiche

Dato che l’implementazione di tutte queste nuove casistiche necessitava la ristrutturazione della funzione di calcolo del danno è stata realizzata una classe ausiliaria aggiuntiva MoveUtilities che contiene le varie funzioni necessarie per l’applicazione del calcolo del danno in maniera più accurata oltre ad altre funzioni di supporto per il calcolo di danni e statistiche.

Inoltre, l’aggiunta di MoveUtilities permette di distinguere fra il calcolo del danno di DoublesMaxDamagePlayer e di DoublesTrueMaxDamagePlayer

SmartPlayer

Questa IA applica in questo ordine i seguenti ragionamenti:

1. Se un Pokémon attivo dell'avversario può essere mandato KO da una mossa di priorità conosciuta dal proprio Pokémon attivo, allora si esegue tale mossa
2. Se il proprio Pokémon attivo conosce la mossa protezione e i Pokémon avversari conoscono mosse super efficaci su di esso, allora si esegue protezione se tale mossa non è stata selezionata nel turno precedente
3. Se il proprio Pokémon attivo rischia di subire un KO da un Pokémon avversario, non ha mosse a disposizione o la mossa ultimocanto (perish song) sta per mandarlo KO, allora effettua uno scambio se esiste un Pokémon scambiabile in grado di resistere all'attacco che si ipotizza verrà effettuato dall'avversario
4. Sceglie la mossa in base ai danni nel caso di mosse di danno o in base agli effetti nel caso di mosse di stato

Fatta eccezione per il punto 1, l'IA genera dei valori random per determinare se effettuare una mossa o meno in maniera tale da rendere l'IA il meno predicibile possibile cercando di mantenere un comportamento il più sensato possibile.

A livello implementativo sono state realizzate più classi ausiliarie:

- La classe MoveHelper per definire la politica di scelta della azione. Attualmente utilizza la logica in figura:

```
def default_choose_command(battle: DoubleBattle, idx_active, last_switch_=_None) -> BattleOrder:
    order: BattleOrder
    order = use_prio(battle, idx_active)

    if order is not None:
        # print("use prio")
        return order

    order = use_protect(battle, idx_active)

    if order is not None:
        # print("use protect")
        return order

    order = should_withdraw(battle, idx_active, last_switch)
    if order is not None:
        # print("use do switch")
        return order

    order = choose_moves(battle, idx_active)
    return order
```

Figura 2 - funzione default_choose_command

- La classe SwitchHelper per poter determinare se effettuare uno scambio di Pokémon attivo e in caso affermativo decide anche con quale Pokémon effettuare lo scambio
- La classe AttackChooser per poter pesare le mosse del singolo Pokémon così da poter determinare quale sia la mossa più conveniente. Inoltre, definisce le funzioni use_prio e use_protect per determinare se il Pokémon attivo deve usare rispettivamente una mossa di priorità o protezione

Statistiche

In questa sezione si mostra la differenza nella performance dei quattro giocatori realizzati in base al numero di vittorie ottenuto su 50 battaglie.

-	rando	elMaxoDamagio	elMaxoDamagioMax	SmartBoyVGC
rando		0.14	0.12	0.22
elMaxoDamagio	0.86		0.02	0.74
elMaxoDamagioMax	0.88	0.98		0.64
SmartBoyVGC	0.78	0.26	0.36	

Corrispondenza tra nome del giocatore e IA:

- **rando**: *RandomPlayer*
- **elMaxoDamagio**: *DoublesMaxDamagePlayer*
- **elMaxoDamagioMax** *DoublesTrueMaxDamagePlayer*
- **SmartBoyVGC** *SmartPlayer*

Lo *SmartPlayer* (**SmartBoyVGC**) risulta relativamente poco efficace contro *DoublesTrueMaxDamagePlayer* (**elMaxoDamagioMax**) e *DoublesMaxDamagePlayer* (**elMaxoDamagio**) perché pensato per funzionare bene contro giocatori umani che agiscono seguendo un ragionamento. Inoltre, il team affidato alle IA è più adatto ad uno stile di gioco aggressivo e perciò più in linea con la mentalità dei due MaxDamagePlayer

Tutti i giocatori che applicano una strategia risultano decisamente migliori del *RandomPlayer* (**rando**).

Sviluppi futuri

Nonostante l'IA faccia uso di buona parte delle meccaniche di gioco, esistono molte situazioni non implementate poiché sono state ritenute o poco impattanti in una normale lotta in doppio o perché avrebbero richiesto ulteriori complicazioni del progetto.

Fra queste ci sono:

- Bacche
- Strumenti tralasciati come magnete (magnet), abilcintura (expert belt)
- Abilità tralasciate come aurafolletto (fairy aura) e pellefolletto (pixilate)
- Bruciapelo (fake out) (non è stato implementato per ora a causa di un baco di poke-env)
- Gli stati veleno e iperavvelenamento
- Condizioni di terreno come "Levitoroccia" (stealth rock) e simili
- Meccaniche generazionali (dinamax, mosse z, megaevoluzioni, terracristallizzazione)
- Sinergie fra Pokémon
- Mosse di redirezione ("sono qui" (follow me) e "polverabbia" (rage powder))
- Altruismo (helping hand)
- Valutare specificamente mosse di self-buff e di debuff (per esempio "danzaspada" (sword dance) e "finte lacrime" (fake tears))

Invece, fra i possibili miglioramenti della logica dell'IA potenzialmente applicabili in uno sviluppo futuro ci sono:

- Valutare diversamente le mosse in base alla propria categoria a seconda se a usarle sia un Pokémon offensivo o di supporto
- Approssimare con maggiore accuratezza gli EV dei Pokémon avversari
- Riconoscere se un Pokémon è tipicamente offensivo o di supporto
- Interfacciarsi con un database per ricavare mosse, statistiche, oggetti e abilità più comuni di un dato Pokémon e usare tali dati per creare una predizione sui Pokémon avversari attivi