Virtual Trainer

`https://github.com/Tensa53/VirtualTrainer`

Daniele Fabiano
Mariantonietta Maselli

Febbraio 2023

Indice

1 Introduzione

Virtual Trainer nasce dall'idea di integrare un modulo intelligente all'interno di un software ben più grande che è quello di SmartGym. SmartGym è il nome del nostro progetto di Ingegneria del Software, il suo obiettivo è quello di dematerializzare e digitalizzare il processo di creazione di una scheda esercizi per la palestra. Virtual Trainer ricopre la funzionalità di generazione di una scheda esercizi. Per questioni logistiche non è stato possibile integrare i due sistemi.

2 Formulazione del problema

Una volta individuato il problema da dover risolvere, sono state effettuate le opportune formalizzazioni che hanno portato alla seguente formulazione:

- **Problema:** Un atleta accede all'applicazione per esercizi chiamata "SmartGym". Decide di creare una scheda esercizi utilizzando la nuova funzionalità recentemente introdotta: Virtual Trainer. L'atleta non dovrà più preoccuparsi della creazione della scheda ed in base alle sue caratteristiche fisiche e di esperienza, l'applicazione ne genererà una in maniera intelligente.
- **Stati:** Ogni esercizio aggiunto/rimosso alla scheda, definisce un cambio di stato.
- **Azioni:** L'aggiunta e la rimozione di un esercizio.
- **Azioni possibili:** $\{add(es.1), add(es.2)...\}, \{remove(es.1), remove(es.3)\}$
- **Modello di transizione:**
  $scheda1\{(es.1), (es.2)\}$
  $risultato(scheda1, add(es.3)) = scheda1\{(es.1), (es.2), (es.3)\}$

  $scheda1\{(es.1), (es.2), (es.3)\}$
  $risultato(scheda1, remove(es.2)) = scheda1\{(es.1), (es.3)\}$
- **Test Obiettivo:** La scheda ha raggiunto il numero prestabilito di esercizi.

3 Specifiche PEAS

- **Performance:** La misura di prestazione si definisce attraverso la misura di affinità degli esercizi e della scheda. La scheda con affinità maggiore è quella migliore. L'agente è performante quando restituisce una scheda con affinità alta.
- **Environment:** L'ambiente del problema è formato dagli esercizi identificati che possono comporre una scheda. Inoltre nell'ambiente è compreso anche l'atleta che desidera usufruire della generazione della scheda.
- **Actuators:** Affinché l'agente possa intraprendere azioni, è necessario che a fronte delle informazioni ricevute su esercizi e atleta, sia in grado di processarle attraverso delle funzioni, i cui risultati influenzeranno l'azione di aggiunta di un esercizio alla scheda.
- **Sensors:** Un archivio di dati dove sono memorizzate le informazioni relative alle proprietà fondamentali degli esercizi per la risoluzione del problema (un nome, la parte del corpo allenata, lo scopo dell'esercizio) e le caratteristiche dell'atleta (peso, altezza, esperienza nella pratica della palestra).

4 Proprietà dell'ambiente

- **Completamente osservabile:** Virtual Trainer conosce in ogni istante lo stato dell'ambiente.
- **Deterministico:** Virtual Trainer sceglierà adeguatamente il prossimo esercizio da inserire nella scheda, sulla base degli esercizi già presenti.
- **Sequenziale:** Virtual Trainer sceglie l'esercizio corrente e questo influenza la scelta del prossimo esercizio.
- **Statico:** Virtual Trainer inizierà le operazioni da effettuare, su un insieme di esercizi che saranno "congelati" e non cambieranno durante il processo.
- **Discreto:** Virtual Trainer potrà scegliere tra un numero di esercizi limitati.
- **Singolo:** Virtual Trainer è l'unico agente coinvolto in questo ambiente.

5 Proof Of Concept

Abbiamo cercato diversi modi di risolvere il problema. In questa sezione saranno brevemente riportate alcune delle tecniche che abbiamo deciso di scartare per questioni di tempi di realizzazione troppo lunghi, per la loro inadeguatezza rispetto al problema e per non averne realmente provato l'ammissibilità.

5.1 Ricerca Tradizionale

Gli algoritmi di ricerca tradizionale intendono risolvere il problema individuando lungo l'albero una sequenza di azioni che formano il cammino che porta allo stato obiettivo. Sia attraverso la ricerca non informata che attraverso quella informata, risultava complicato individuare una corretta sequenza. Analizzando il nostro problema, non siamo riusciti a definire la soluzione come un cammino verso l'obiettivo. Per questo motivo abbiamo deciso di scartare la ricerca tradizionale.

5.2 Ricerca con Avversari

Gli algoritmi di ricerca con avversari sono un particolare tipo di algoritmi di ricerca in profondità, che lavorano sull'albero di gioco in un ambiente multi-agente. L'idea era quella di creare il gioco della "battaglia all'ultimo muscolo": due fasce muscolari adatte ad essere allenate insieme diventano i nostri due giocatori. Per ogni livello dell'albero, era definito il turno del relativo giocatore che avrebbe scelto come mossa quella che lo avrebbe indirizzato verso l'esercizio migliore. La difficoltà in questo caso era legata al non essere riusciti a trovare delle regole al gioco. Inoltre il gioco si sarebbe potuto sviluppare solamente come un gioco con numero di turni prestabiliti dove non erano ben specificate delle mosse killer che avrebbero indirizzato il gioco verso lo stato terminale e quindi alla sua fine. Per questo motivo abbiamo deciso di scartare la ricerca con avversari.

5.3 Classificazione

La classificazione è un particolare task che permette di individuare il valore di una variabile dipendente categorica mediante le variabili indipendenti. Si intendeva realizzare un classificatore che sulla base dei valori delle variabili indipendenti ***peso*** e ***altezza***, avrebbe stabilito il valore corretto della

variabile dipendente categorica ***Indice di Massa Corporea***. Dovendo risolvere il problema attraverso tecniche di Machine Learning, i costi di tempo hanno impedito l'approfondimento di queste soluzioni perchè era necessario dover costruire il dataset e l'intera fase di ingegnerizzazione avrebbe impiegato troppo tempo.

5.4 Clustering

Il clustering è un particolare task che suddivide gli oggetti con caratteristiche simili all'interno di gruppi molto omogenei al loro ma che tra loro sono eterogenei. In questo caso si voleva tentare di raggruppare gli esercizi su una base di caratteristiche con valori tra di loro simili. Sulla base delle informazioni inserite dall'utente, sarebbero stati scelti gli esercizi del relativo cluster ottenuto precedentemente. Le motivazioni che non ci hanno portato ad effettuare il clustering, sono le medesime illustrate per la classificazione.

6 Una possibile soluzione

6.1 Gli algoritmi genetici

Gli algoritmi genetici sono strumenti di intelligenza artificiale ispirati alla teoria dell'evoluzione di Darwin: la selezione naturale, l'adattamento e la teoria della sopravvivenza vengono utilizzati per risolvere problemi di ottimizzazione computazionalmente difficili. Lavorano evolvendo le soluzioni candidate e producendo di volta in volta individui sempre migliori rispetto ad una funzione obiettivo, fino a raggiungere l'ottimo o un'altra condizione di terminazione. Abbiamo optato per questa tipologia di algoritmi per una serie di motivazioni:

- **Tecnica di ricerca veloce:** esplorano rapidamente lo spazio di ricerca e con una buona configurazione vengono prodotti risultati vicini all'ottimo in tempi ragionevoli.

- **Sono una valida scelta per problemi multiobiettivo:** nel nostro caso, sarà la media di due funzioni a determinare l'adeguatezza degli individui.

6.2 Codifica degli individui

Gli algoritmi genetici codificano gli individui come stringhe di lunghezza finita. Abbiamo deciso di codificare un individuo come un array di interi, ogni intero fa riferimento all'identificativo univoco di ciascun esercizio. La codifica degli individui indicherà le soluzioni candidate.

6.3 Inizializzazione della popolazione

Sarà data all'atleta la possibilità di specificare una o più parti del corpo da allenare, la popolazione sarà inizializzata scegliendo tra quegli esercizi che allenano le parti del corpo indicate. Nel caso in cui non siano specificate parti del corpo, la popolazione sarà inizializzata facendo riferimento all'intero set di esercizi disponibili.

6.4 Funzione di fitness

Come accennato, ci troviamo di fronte a un problema multiobiettivo. La funzione di fitness che determina la bontà degli individui è definita come un vettore di due fitness, in particolare si avrà:

$$fitness(s) = \langle Avg(affinità), Avg(sfida) \rangle$$

Dove ***s*** rappresenta la scheda esercizi.
Verranno ora definite in dettaglio:

6.4.1 Funzione di affinità

La prima funzione che verrà calcolata lavora sull'***affinità*** dell'esercizio rispetto al traguardo dell'atleta.

Ogni esercizio ha una sua tipologia ***t***:

- **Aerobico:** camminata, corsa, saltare la corda...
- **Anaerobico:** piegamenti, trazioni, squat...

Il traguardo è stabilito attraverso l'***Indice di Massa Corporea*** dell'atleta, calcolabile come $peso/altezza^2$. In particolare:

- $16 \leq IMC < 19$: L'atleta è sottopeso, ha bisogno di aumentare la massa muscolare per raggiungere il normopeso.
- $19 \leq IMC < 25$: L'atleta è già normopeso ma potrebbe raggiungere il peso forma. A seconda del valore IMC distinguiamo tre casi:
  - $19 \leq IMC < 20$: L'atleta è tendente al sottopeso, ha bisogno di aumentare la massa muscolare per raggiungere il peso forma.
  - $20 \leq IMC \leq 24$: L'atleta è nel peso forma, ha bisogno di tonificare per mantenere il peso forma.
  - $24 < IMC \leq 25$: L'atleta è tendente al sovrappeso, ha bisogno di dimagrire per raggiungere il peso forma.
- $25 < IMC \leq 30$: L'atleta è sovrappeso, ha bisogno di dimagrire per raggiungere il normopeso.

Si assume che gli utilizzatori di questo strumento siano persone in salute, per questa ragione i casi limite come grave magrezza e obesità non sono considerati nell'analisi del problema.

Consideriamo un atleta $\boldsymbol{a}$ ed un esercizio $\boldsymbol{e}$, la funzione di affinità è definita come:

- affinità(a,e) = 100 nei seguenti casi:
  - (16 ≤ a.IMC < 19) AND (e.t = "Aerobico")
  - (25 < a.IMC ≤ 30) AND (e.t = "Anaerobico")
- affinità(a,e) = 200 nei seguenti casi:
  - (19 ≤ a.IMC < 20) AND (e.t = "Aerobico")
  - (24 < a.IMC ≤ 25) AND (e.t = "Anaerobico")
- affinità(a,e) = 300 nei seguenti casi:
  - (16 ≤ a.IMC < 19) AND (e.t = "Anaerobico")
  - (19 ≤ a.IMC < 20) AND (e.t = "Anaerobico")
  - (20 ≤ a.IMC ≤ 24) AND (e.t = "Anaerobico")

- $(20 \leq$ a.IMC $\leq 24)$ AND (e.t = "Aerobico")
- $(24 <$ a.IMC $\leq 25)$ AND (e.t = "Aerobico")
- $(25 \leq$ a.IMC $\leq 30)$ AND (e.t = "Aerobico")

La fitness della scheda esercizi $\boldsymbol{s}$ è ottenuta come una media delle fitness degli $\boldsymbol{n}$ esercizi che la compongono, quindi calcoliamo:

$$Avg(affinità) = \frac{1}{n} \sum affinità(a, e)$$

6.4.2 Funzione di sfida

La seconda funzione che verrà calcolata lavora sulla ***sfida*** per l'atleta.
Dobbiamo determinare il livello di esperienza dell'atleta. Abbiamo deciso di rappresentarlo come segue:

$$esperienza(a) = 120 + (m * 10)$$

Dove $\boldsymbol{a}$ rappresenta l'atleta e $\boldsymbol{m}$ rappresenta il numero di mesi in cui l'atleta ha precedentemente effettuato palestra.
Così facendo, dividiamo i livelli di esperienza $\boldsymbol{e}$ in quattro categorie:

- **Principiante:** $120 \leq e \leq 360$
- **Intermedio:** $370 \leq e \leq 600$
- **Esperto:** $610 \leq e \leq 840$
- **Massimo esperto:** $e > 840$

Abbiamo bisogno di quantificare anche la difficoltà intrinseca di ciascun esercizio, la calcoliamo come la media tra il valore minimo e il valore massimo di esperienza per i primi tre livelli. Ciascun esercizio potrà quindi essere:

- **Facile:** $d = (120 + 360)/2 = 240$
- **Medio:** $d = (370 + 600)/2 = 485$
- **Difficile:** $d = (610 + 840)/2 = 725$

Dove ***d*** rappresenta la difficoltà dell'esercizio.
La difficoltà dell'esercizio è indipendente dai parametri personalizzabili dall'utente. L'atleta potrà variare a suo piacimento il numero di ripetizioni (espresso come un intero) o la durata dell'esercizio (espresssa in secondi).

La funzione che dobbiamo massimizzare è la seguente:

$$sfida(a, e) = SFIDAMAX - |e - d|$$

La fitness della scheda esercizi ***s*** è ottenuta come una media delle fitness degli ***n*** esercizi che la compongono, quindi calcoliamo:

$$Avg(sfida) = \frac{1}{n} \sum sfida(a, e)$$

6.5 Vincoli e stopping condition

6.5.1 Preference sorting

Ci troviamo di fronte a un vettore di funzioni di fitness, il fronte di Pareto e il concetto di preference sorting sono molto utili per aiutarci a comprendere e selezionare le soluzioni ottimali tra molte opzioni.
Il fronte di Pareto rappresenta tutte le soluzioni ottimali non dominate. Una soluzione è considerata non dominata se non esiste un'altra soluzione che sia migliore in tutti i criteri di valutazione e allo stesso tempo non peggiore in almeno un criterio. Il concetto di preference sorting si riferisce alla capacità di classificare le soluzioni ottenute da un algoritmo di ottimizzazione multi-obiettivo in base alle preferenze specificate. Effettueremo il preference sorting rispetto alla funzione di ***affinità***. Gli esercizi con una sfida non ottimale, in questo caso verranno considerati come esercizi "di recupero" per l'atleta.

6.5.2 Vincoli del problema

A metà della generazione della scheda, l'algoritmo si fermerà per chiedere maggiori informazioni all'utente su come proseguire (se rimuovere esercizi, se concentrarsi su una parte del corpo specifica...). Al termine di ogni iterazione dell'algoritmo verrà presentata all'utente la scheda e gli verrà chiesto se intende conservare la scheda attuale o la precedente, in ciascuno dei casi l'algoritmo verrà interrotto. Nel caso in cui l'utente decida di far generare un'altra scheda, l'algoritmo continuerà la sua esecuzione.

6.5.3 Stopping condition

Le stopping conditions sono condizioni predefinite che indicano quando un algoritmo genetico deve interrompersi. L'algoritmo si ferma dopo un certo numero di generazioni determinate dall'utente a seconda del suo grado di gradimento della scheda generata e, come spiegato nella sezione precedente, ci sarà uno stop intermedio sul numero di esercizi inseriti per prelevare ulteriori informazioni e migliorare la generazione della scheda.

6.6 Operatori genetici

Selezione, ***crossover*** e ***mutazione*** sono i tre principali operatori utilizzati negli algoritmi genetici per generare nuove soluzioni e mantenere la varianza nella popolazione di soluzioni. La selezione è il processo di selezione di individui dalla popolazione corrente per formare la prossima generazione. Il crossover è un processo di incrocio che prende due individui dalla popolazione e crea un nuovo individuo combinando le loro informazioni genetiche. La mutazione è un processo che modifica casualmente le informazioni genetiche di un individuo. L'algoritmo migliore per ogni operatore genetico sarà stabilito mediante la sperimentazione empirica.

6.6.1 Risultati della sperimentazione empirica

Per effettuare la sperimentazione empirica, abbiamo scelto alcune tipologie di algoritmi degli operatori per ottenere diverse configurazioni dell'algoritmo genetico. Per ogni operatore, sono stati scelti i seguenti algoritmi:

- **Selezione:**
  1) *3-way tournment*, ogni elemento viene confrontato con k-1 elementi in una serie di confronti individuali. Il vincitore di ogni confronto viene segnato e il processo viene ripetuto fino a quando non viene identificato il miglior elemento.
  2) *Truncation*, vengono selezionati i migliori elementi della popolazione attraverso una rigida valutazione sugli elementi con fitness migliore.

- **Crossover:**
  3) *One point crossover*, i genitori vengono rappresentati come stringhe di codice genetico e il punto di crossover viene selezionato casualmente.

Una volta selezionato il punto di crossover, i geni dei due genitori vengono scambiati in modo da creare due nuovi individui.
4) *Two point crossover*, sono utilizzati due punti di crossover.

- **Mutazione:**
  5) *Random Resetting*, mutiamo un individuo scegliendo casualmente un nuovo valore per ogni gene presente nella stringa di codice genetico dell'individuo.

Abbiamo effettuato 5 esecuzioni dell'algoritmo e calcolato la media delle funzioni di affinità e sfida per le diverse configurazioni, i risultati sono stati i seguenti:

- **Configurazione (1, 4, 5):**
  $AVG(affinità(s)) = 273.33$, $AVG(sfida(s)) = 428.66$
- **Configurazione (2, 4, 5):**
  $AVG(affinità(s)) = 286.66$, $AVG(sfida(s)) = 533.83$
- **Configurazione (1, 3, 5):**
  $AVG(affinità(s)) = 273.32$, $AVG(sfida(s)) = 477.33$
- **Configurazione (2, 3, 5):**
  $AVG(affinità(s)) = 273.32$, $AVG(sfida(s)) = 509.40$

La **Configurazione(2, 4, 5)** si è rivelata la migliore, per cui abbiamo optato per questi operatori genetici.

7 Implementazione

Stabilita la tecnica di risoluzione, si è poi passati alla fase di implementazione dell'algoritmo in un linguaggio di programmazione. L'attuale implementazione è una versione semplificata dell'algoritmo che non tiene conto dei vincoli relativi all'interattività con l'utente. L'utente interagirà con l'algoritmo esclusivamente per inserire le sue informazioni.

7.1 Strumenti usati

Per realizzare l'algoritmo abbiamo fatto uso di Python ed in particolare del framework DEAP, che fornisce dei semplici algoritmi di riferimento che possono essere poi modificati e personalizzati. Inoltre permette in maniera molto rapida il cambio di configurazione degli operatori dell'algoritmo.

7.2 Il codice in breve

Il codice scritto è stato organizzato nel seguente modo:

- **Inserimento e modifica informazioni atleta:** L'utente può inserire le informazioni relative a *peso, altezza, mesiEsperienza* che potranno poi essere successivamente modificate.
- **Calcolo caratteristiche atleta:** Sulla base delle informazioni dell'atleta, saranno calcolate le sue caratteristiche (*IMC, Categoria, Esperienza e livello di esperienza, Traguardo da raggiungere*). Le caratteristiche saranno nuovamente calcolate quando saranno modificate le informazioni.
- **Configurazione parametri dell'algoritmo** Saranno settati tutti i parametri dell'algoritmo, viene quindi stabilita la tipologia di fitness, la codifica degli individui e gli operatori genetici utilizzati.
- **Esecuzione dell'algoritmo** Una volta settati i parametri l'algoritmo verrà lanciato e restituirà il miglior individuo dell'esecuzione.

8 That's all folks!

8.1 Curiosità

Il quadretto di immagini presenti nella prima pagina è stato generato attraverso uno strumento di Deep Learning chiamato DALL-E 2 di OpenAI.

8.2 Conclusioni

Questo progetto ci ha aiutato a capire meglio il funzionamento degli algoritmi genetici. Attraverso la messa in pratica dell'algoritmo siamo stati in grado di produrre del codice di implementazione, rendendo più concreto tutto ciò che abbiamo studiato in questi mesi.