

Progetto Fondamenti di Intelligenza Artificiale

2020/2021

|  |  |
| --- | --- |
| Felice De Chiara | 0512105866 |
| Vincenzo Emanuele Martone | 0512105758 |
| Antonio Russo | 0512106058 |
| Alfonso Graziano | 0512105776 |

## Definizione del problema

Sito che permette all'utente di navigare tra le schede tecniche di vari cellulari, e che dispone di un tool che consiglia un cellulare all'utente, sulla base di un budget e sulla base di quanto all'utente importino determinate caratteristiche (ad esempio l'utente dovrà dire su una scala da 1 a 10 quanto gli importano batteria, fotocamera, display e prestazioni). Il dataset è facilmente reperibile grazie ad un semplice tool di scraping. Per quanto riguarda il metro di paragone per confrontare le caratteristiche dei cellulari, abbiamo pensato di introdurre un sistema di punteggi che i recensori (che nel nostro dominio applicativo sono un team di professionisti) assegnano alle caratteristiche stesse e sono dunque parte integrante della scheda tecnica del prodotto; in caso di pareggio su determinate caratteristiche, si terrà conto delle recensioni degli utenti.

## Specifica dell’ambiente

* Performance
  + L’algoritmo considera i seguenti parametri:
    - Minimizzare lo scarto di:
      * Battery
      * Performance
      * Display
      * Camera
    - Massimizzare la data di uscita
      * Es: a parità di altri parametri, uno smartphone uscito nel 2019 sarà scelto al posto di uno uscito nel 2016
* Environment
  + Completamente osservabile
    - L’agente conosce in ogni momento tutto lo spazio degli stati
    - Nello specifico in ogni momento tiene in memoria la lista delle specifiche degli smartphone
  + Agente singolo
  + Deterministico
    - Lo stato è deterministico in quanto non ci sono azioni esterne che influenzano il prossimo stato. Lo stato successivo infatti deriva dall’azione e dallo stato corrente
  + Discreto
    - Il numero di percezioni dell’agente è limitato
    - Nel nostro caso abbiamo un numero discreto di specifiche, azioni e percezioni
  + Statico
    - L’ambiente è statico in quanto l’agente elabora su un’istanza dell’insieme delle schede tecniche aggiornate fino a quel momento
  + Episodico
* Actuators
  + Mostrare a video un set di “specs” conformi
* Sensors
  + Input di campi => range float da 1 a 10
  + Database
    - Noi leggiamo i dati da un form e il sistema manda il dataset dal db

## Soluzione

Per risolvere questo problema di ricerca abbiamo valutato diverse opzioni. L’opzione più promettente ci è sembrata quella di utilizzare un **algoritmo genetico** proprio perché le sue caratteristiche sono simili alle caratteristiche del problema.

Il nostro obiettivo infatti è quello di restituire un insieme di smartphone che meglio approssimano le caratteristiche richieste in input dall’utente. Nello specifico le caratteristiche sono: **battery, performance, display, camera**.

Pertanto un **individuo** non è altro che un insieme di schede tecniche (banalmente un array formato da n schede tecniche).

## Il confronto con OPT

Durante lo sviluppo dell’algoritmo genetico abbiamo cercato un modo per stabilire quanto fosse buona una configurazione di parametri. Abbiamo quindi pensato che nel caso specifico ci è possibile calcolare la soluzione ottima senza troppe difficoltà dato che il dataset si presta per un’operazione di ricerca lineare.

### Come trovare la soluzione ottima?

La soluzione ottima è composta da un individuo che minimizza lo scarto rispetto ai parametri presi in input. Sarà quindi composta dalle x schede tecniche che più si avvicinano a quei parametri.   
Procediamo quindi scorrendo tutta la lista e calcoliamo il **valore di similitudine** in base ai parametri e ordiniamo la lista in maniera non decrescente basandoci su questo valore. La soluzione ottima (OPT) è formata dal set dei primi x individui della lista ottenuta.

### Come valutare una configurazione dell’algoritmo genetico con **OPT**?

Data una configurazione dell’algoritmo genetico, lo eseguiamo un numero y di volte. Da ogni iterazione otteniamo un valore minimo di fit (in base ai parametri passati) e al termine ne effettuiamo la **media**.

Questa media verrà confrontata con il valore di fit di OPT ottenuto sommando tutti i valori di fit delle schede tecniche all’interno del set di OPT.

Per calcolare il grado di ottimalità di una determinata configurazione dell’algoritmo genetico, non facciamo altro che valutare la sua vicinanza al valore di OPT.

Es. OPT = 3.12 | G1 = 4.7   
La vicinanza è G1-OPT = 4.7-3.12 = 1.58

OPT = 3.12 | G2 = 5.3   
La vicinanza è G2-OPT = 5.3-3.12 = 2.18

Viene preferita la configurazione G1, in quanto lo scarto è quello che più si avvicina a OPT

Nella prossima sezione del documento analizzeremo tutti i parametri utilizzati e tutti i test effettuati

## Parametri

### Selezione degli individui

Al momento l’unico algoritmo di selezione utilizzato è il Truncation. Vengono scelti i primi n individui ordinati in base alla minimizzazione della funzione di fitting.

Per aggiungere una componente di casualità eventualmente si potrebbe valutare l’utilizzo di un algoritmo Roulette Wheel

### Mutazione

La mutazione avviene tramite Random Resetting. All’interno dell’individuo viene iniettato una scheda tecnica randomica in una posizione random dopo aver effettuato il crossover

### Crossover

Il crossover può essere di diversi tipi. Nell’algoritmo genetico al momento è implementato un crossover di tipo Single Point.

### Funzione di fitness

Originariamente il problema era pensato come multiobiettivo. Vogliamo infatti minimizzare lo scarto di tutti i parametri. Per semplificare abbiamo utilizzato un whole approach per cui abbiamo fuso tutte le funzioni in una singola funzione di fitness.

La prima versione della funzione di fitness era una semplice somma lineare di tutti i parametri (performance, battery, camera, display). Una somma lineare è fallimentare in quanto non scarta abbastanza velocemente i risultati più “scarsi”.

F(n) = scartoPerformance + scartoBattery + scartoCamera + scartoDisplay

Abbiamo poi pensato ad una somma di tutti i parametri a cui moltiplichiamo una costante:  
F(n) = scartoPerformance\*c + scartoBattery\*c + scartoCamera\*c + scartoDisplay\*c

A questo punto la funzione di fitness approssimava meglio ma ancora i risultati non venivano divisi abbastanza bene. Abbiamo quindi utilizzato una funzione esponenziale:

F(n) = scartoPerformance^d + scartoBattery^d + scartoCamera^d + scartoDisplay^d

Questo ha portato ad un bug in quanto per valori di scarto minori di 1 il quadrato è minore dello scarto (es 0.2^2 = 0.04). Abbiamo risolto quindi rendendo la funzione composta. La funzione al momento è quindi la seguente:   
scarto = scartoPerformance>1 ? scartoPerformance\*c : scartoPerformance^d

Una volta fatte queste considerazioni abbiamo riflettuto sull’aggiungere un’altra misura importante: la data d’uscita. Nello specifico abbiamo a disposizione il mese e l’anno di uscita del device. Procediamo quindi a sottrarre la data attuale da quella d’uscita per sapere da quanti mesi è uscito lo smartphone.

Anche in questo caso bisogna capire come integrare questa nuova variabile all’interno della funzione di fitting per implementare l’aging. A parità di prestazioni infatti, un dispositivo uscito qualche anno fa oggi ha sicuramente prestazioni inferiori.

Avevamo pensato ad utilizzare il preference sorting per fare sorting in base alla data d’uscita ma aggiungere quella variabile all’interno della funzione di fitting ci ha reso più semplice il calcolo

Proprio in merito all’inserimento della componente “Data” all’interno della funzione di fitting, abbiamo notato un degrado dei risultati dell’algoritmo nel momento in cui è stata implementata, in quanto l’algoritmo vincolava in maniera troppo stretta i risultati alla data d’uscita. Ci siamo, dunque, resi conto del fatto che fosse necessario capire quale fosse il peso da dare alla “Data” sulla scrematura dei nostri risultati.

### Stopping condition

Abbiamo testato diverse stopping condition. Alla fine abbiamo optato per monitorare il tempo di esecuzione e stoppare l’algoritmo dopo 1 secondo. Dato il numero molto limitato presente nel dataset (al più circa un migliaio di device) l’algoritmo arriva molto velocemente a convergenza.

Abbiamo scelto questo parametro anche basandoci su un requisito non funzionale all’interno del progetto di Ingegneria del Software per cui il tool DoraIA deve ritornare un valore entro max 5 secondi.

### Passi dell’algoritmo

1. Costruire una popolazione iniziale contenente tutti gli individui del DataSet
2. Calcolare il loro valore di fitness
3. Selezionare un sottoinsieme di individui della popolazione iniziale per ammetterli nel Mating Pool tramite Truncation
4. Gli individui precedentemente selezionati vengono fatti accoppiare tramite Single Point Crossover
5. Gli individui ottenuti dal Crossover vengono mutati tramite una Random Resetting e si ritorna al passo 3

## Testing dei parametri scelti