EVOLUTIONARY ALGORITHM

A CURA DI LINO POLO

Idea di fondo

La computazione evolutiva comprende una grande collezione di tecniche ispirate, in linea molto generale, alle leggi dell'evoluzione naturale proposte dal biologo, naturalista, antropologo ed esploratore Charles Darwin (1809-1882). In tal senso, gli individui meglio preparati sono quelli che hanno un indice di sopravvivenza migliore e, quindi, fanno più figli. La teoria di Jean-Baptiste Lamarck (1744-1829), al contrario di quella darwiniana, si basa sull'ereditarietà dei caratteri acquisiti, cioè sulla capacità degli individui di trasferire alla discendenza gli adattamenti all'ambiente messi a punto in vita.

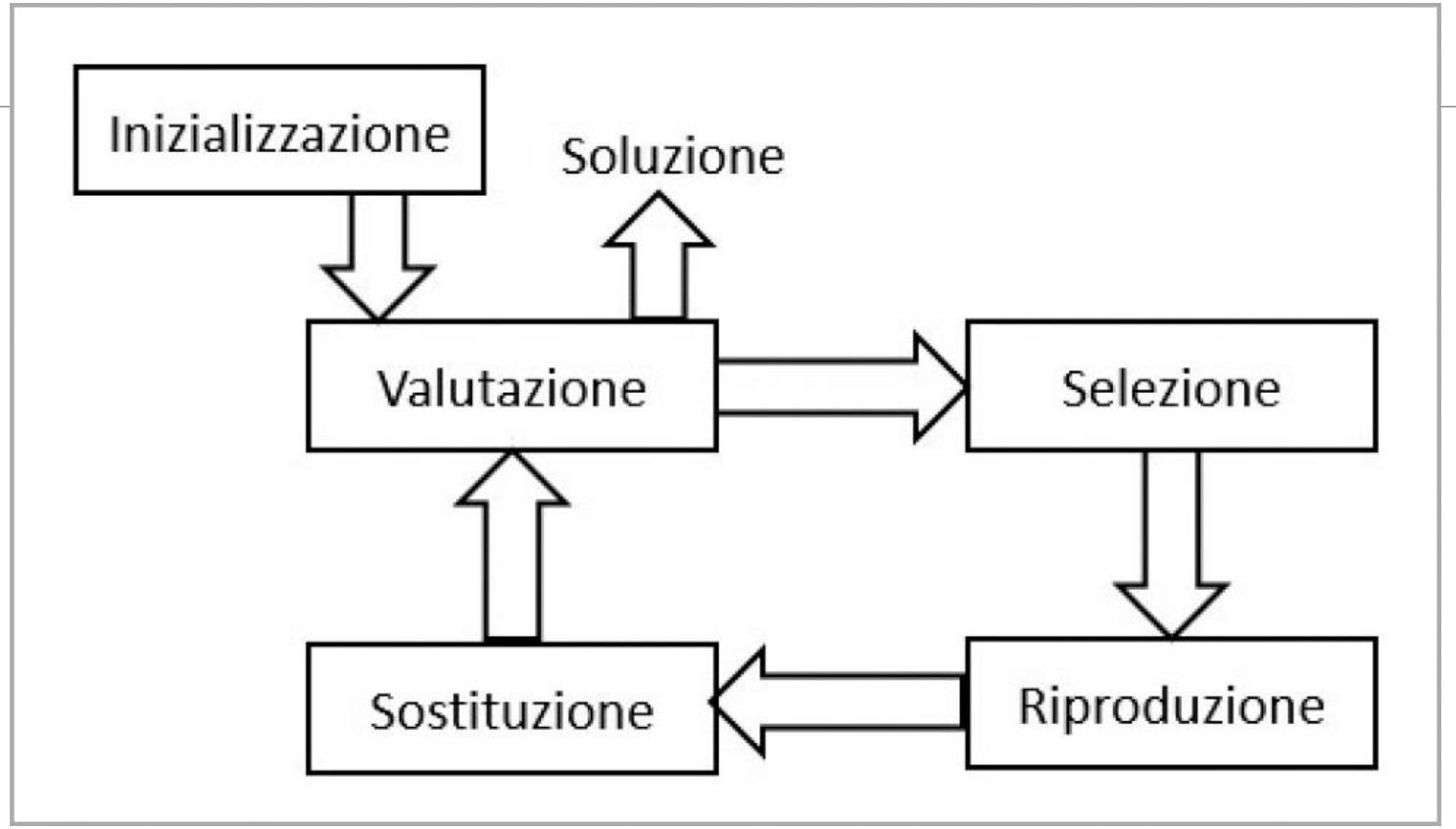
In sostanza, come un individuo di una popolazione di organismi deve adattarsi all'ambiente che lo circonda per sopravvivere e riprodursi, così una possibile soluzione deve essere adatta a risolvere il problema al quale è rivolta.

Meccanismo di funzionamento

L’evoluzione, naturale o artificiale, non ha nulla di intelligente nel senso definito nel Capitolo 1, non capisce quello che sta facendo. L’intelligenza può essere vista come un fenomeno emergente dall’evoluzione, nel senso che gli organismi prodotti possono rappresentare una soluzione migliore rispetto a quelli iniziali, ed essere perciò dotati di una qualche forma di intelligenza. Per questo motivo, gli algoritmi evolutivi rientrano nell’AI.

Nel contesto AI, si parla di ambiente come del problema da risolvere, mentre la popolazione riguarda gli individui che vivono nell’ambiente, ciascuno dei quali rappresenta una possibile soluzione al problema. Questi algoritmi valutano la bontà di ogni soluzione, denominata fitness, che sia buona o cattiva, per selezionare le migliori da cui creare una seconda generazione di individui.

Il processo iterativo rappresentato in Figura vuole mostrare l’evoluzione degli individui nelle generazioni. Gli individui sono valutati, selezionati e incrociati per ottenere nuove generazioni ogni volta. I criteri di arresto, dipendenti dal problema, fermano questo processo e forniscono una soluzione.


Inizializzazione
Soluzione
Valutazione
Selezione
Sostituzione
Riproduzione

Esistono vari tipi di algoritmo evolutivo; la differenza consiste nei vari modi in cui si realizza ognuna delle fasi mostrate in figura e descritte nei sottoparagrafi. Per comprendere meglio come funziona l'approccio evolutivo si può seguire l'esempio mostrato nel paragrafo sugli algoritmi genetici.

Essi vengono impiegati per risolvere i problemi di ricerca globale, perché consentono di esplorare una gamma di soluzioni potenziali di gran lunga più ampia di quella che si può esaminare con i programmi convenzionali. Come svantaggio operativo, presentano difficoltà di progettazione dovute ai parametri da impostare e alla necessità di utilizzare una popolazione elevata.

Inizializzazione

Questa fase dipende soprattutto dal tipo di problema da risolvere, con attenzione ai vincoli imposti e ai valori ammessi. Si possono usare gli approcci descritti nel Capitolo 9 per adeguare i dati da trattare.

Si prepara la popolazione delle soluzioni, in maniera più o meno casuale secondo i vincoli, cercando di avere una certa diversità, in modo da essere sicuri di non lasciare nessuno spazio da esplorare.

Il modo di rappresentare la conoscenza in una soluzione è molto importante, perché determina il funzionamento di questo paradigma.

Valutazione

Si tratta della fase più importante dell'algoritmo, perché definisce il problema da risolvere. Bisogna anche stabilire la condizione di termine con cui finire l'evoluzione, altrimenti non si finirebbe mai.

Si comincia con la ricostruzione della soluzione, poi si calcola la validità assegnando a ognuna un valore di bontà, di efficacia o di prestazione. Nelle successive evoluzioni questo valore viene usato per distinguere tra soluzioni buone e inefficienti. Il meccanismo usato può essere lento, complicato e fonte di errore, tanto che la stessa soluzione viene valutata più volte producendo diversi gradi di bontà. L'errore aumenta quando non si riesce a riprodurre ogni aspetto del problema.

Selezione

Dopo aver valutato tutti gli individui della popolazione, si passa a scegliere i migliori che si riprodurranno per creare la prossima generazione. È la fase in cui avviene la cosiddetta evoluzione naturale. Si può regolare in maniera più o meno forte, per creare una generazione successiva più o meno popolosa. Se si vuole ridurre il tempo di calcolo e arrivare subito a qualcosa di utile, si può scegliere di creare una selezione forte per avere pochi individui. Ma c'è il rischio di perdere possibilità di esplorazione e di finire in posizioni di massimo locale della funzione da ottimizzare.

Esistono diverse strategie per selezionare gli individui migliori da riprodurre introducendo un po' di diversità per tentare nuove strade. In tal modo, una soluzione, per quanto sia fatta male, avrà sempre qualche probabilità di essere selezionata anche se la popolazione comprende soluzioni molto migliori.

Riproduzione

Dopo aver selezionato gli individui da portare nella prossima generazione, si passa alla fase di riproduzione per poterli generare.

Un certo algoritmo evolutivo prende il suo nome proprio dal tipo di riproduzione adottato. Infatti, gli algoritmi genetici, spiegati nel prossimo paragrafo, sono algoritmi evolutivi in cui la riproduzione è svolta con la mutazione.

Sostituzione

Il ciclo si chiude con la fase di sostituzione per scegliere quali individui della generazione precedente saranno sostituiti da nuovi individui generati con la riproduzione. In genere, si lascia l'individuo migliore e si sostituiscono tutti gli altri.

In questo modulo si possono applicare le tecniche di speciazione, cioè metodi che facilitano l'identificazione di soluzioni diverse per quei problemi che hanno diverse posizioni ottimali. Un metodo di sostituzione è denominato niching. Si tratta di prevedere la selezione, per ogni nuovo individuo generato, degli individui della popolazione precedente che più si avvicinano a esso. Così, nella generazione successiva, solo l'individuo migliore del gruppo di simili potrà sopravvivere.

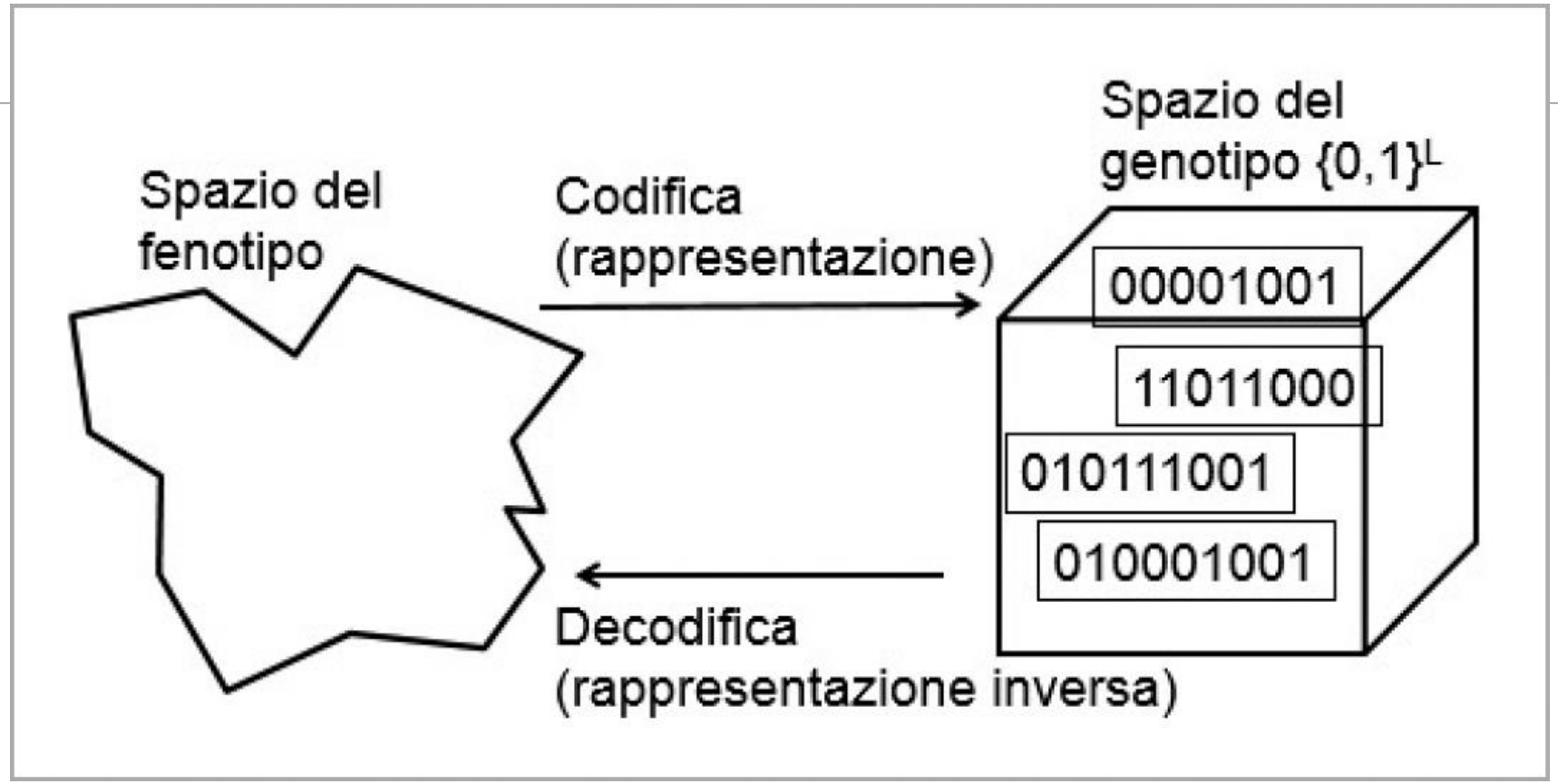
Genetic algorithm

I genetic algorithm (algoritmi genetici) sono la tipologia più conosciuta di algoritmi evolutivi. Si ispirano, in maniera molto generale, alla genetica, la branca della biologia che studia i geni, l'ereditarietà e la variabilità genetica negli organismi viventi. Si tratta di una tecnica stocastica di ottimizzazione che procede in modo iterativo. Gli operatori disponibili riguardano riproduzione, ricombinazione e mutazione.

Il phenotype (fenotipo) è la manifestazione dei caratteri, per esempio il colore degli occhi di un individuo. Il genotype (genotipo, cromosoma) del singolo individuo, rappresentato con una stringa binaria di lunghezza fissa l, è la caratteristica principale di questo approccio. Il cromosoma codifica il genotipo con una stringa di variabile. Il gene è la variabile facente parte della stringa di codifica. Allele è il valore della singola variabile nel gene costituente il cromosoma.

Pertanto, si può effettuare una mappatura tra genotipo e fenotipo. Cioè, per ogni individuo si considera l'informazione del suo cromosoma (genotipo) per simulare la soluzione rappresentata (fenotipo). Nel contesto digitale ogni cromosoma può avere solo geni con valori del bit 0 o 1, così un genotipo viene costruito con un cromosoma corrispondente a una sequenza di bit.

La popolazione contiene l'insieme di tutti i genotipi, è lo spazio di ricerca in cui cercare la soluzione; questo insieme di stringhe ha 2l elementi perché si tratta di numeri binari. La Figura 5.2 presenta un esempio di codifica da fenotipo a genotipo per ottenere lo spazio in cui operare con l'algoritmo genetico, seguita dalla decodifica per ricostruire il fenotipo ottenuto dalla soluzione.


Spazio del
fenotipo
Codifica
(rappresentazione)
Spazio del
genotipo {0,1}L
00001001
11011000
010111001
010001001
Decodifica
(rappresentazione inversa)

Gli algoritmi genetici possono fornire soluzioni interessanti per questi motivi:

esplorazione: eseguono una ricerca su aree promettenti dello spazio di ricerca;

sfruttamento: ottimizzano in un'area promettente;

sussiste co-operazione e competizione tra loro;

il crossover è esplorativo, esegue un grosso salto in un'area tra due aree denominate come genitore;

la mutazione crea piccole diversificazioni, perciò rimane vicino all'area del genitore.

Ovviamente, questi algoritmi non si rendono conto di avvicinarsi alla soluzione del problema, hanno sempre bisogno della guida dell'uomo.

Algoritmo di creazione

I parametri da gestire per un algoritmo genetico sono:

dimensione della popolazione;

criterio di arresto: numero massimo di iterazioni, soglia di fitness;

distribuzione operatori per la nuova generazione:

- probabilità di clonazione;
- probabilità e tipo di crossover;
- probabilità di mutazione.

Per creare un algoritmo genetico bisogna eseguire questi passi:

1. inizializzazione: genera la popolazione iniziale di soluzioni;

2. valutazione: valuta la fitness di ogni soluzione;

3. selezione, finché la condizione di termine non è raggiunta esegui:

a. seleziona le soluzioni per la riproduzione;

b. riproduzione: ricombina le soluzioni selezionate;

c. mutazione delle soluzioni;

d. valuta la fitness delle soluzioni modificate;

e. sostituzione: genera una nuova popolazione rimpiazzando la popolazione iniziale con le soluzioni modificate;

4. fine.

Nei prossimi sottoparagrafi vengono dettagliate le operazioni da eseguire; per comprendere meglio cosa bisogna fare si possono seguire i commenti nel codice mostrato nel paragrafo “Libreria DEAP”.

Per esempio, considerare il percorso del commesso viaggiatore, noto come Travelling Salesman Problem (TSP), in cui il commesso viaggiatore deve attraversare una serie di città cercando di ridurre le spese vuole usare un software che gli dica il percorso da fare. L'algoritmo genetico fornisce solo l'ordine delle città da visitare, non è necessario inserire il suo codice nel software che fornisce i percorsi, basta dargli soltanto i risultati. Il commesso segue l'ordine fornito dal software ogni volta che deve fare quel tipo di percorso; se cambiano le città bisogna rifare tutti i calcoli, l'algoritmo genetico fornisce un nuovo percorso che va ad aggiornare il software di gestione per fargli dare un nuovo ordine al commesso.

Rapporti con altri modelli

Controllo fuzzy

Tali modelli si possono usare per la taratura dei parametri di un sistema di controllo gestito con la fuzzy logic, descritta nel Capitolo 7. In tal caso, si fissa un insieme di parametri che definiscono la forma delle funzioni di appartenenza del meccanismo di inferenza fuzzy con cui si forma il cromosoma. La funzione di fitness viene scelta secondo il tipo di controllo da gestire. Si controllano i valori assunti per evitare situazioni senza significato fisico nell'ambito del controllo da gestire. Due svantaggi derivano dal tempo di calcolo, che può diventare troppo lungo, e dalla riduzione della trasparenza che si può rivelare applicando il risultato. Un articolo con le basi dell'approccio evolutivo è quello di Karr (1991).

La fuzzy logic viene usata anche per aiutare gli algoritmi evolutivi. Si è dato il nome di "governo fuzzy" alle regole fuzzy incaricate di controllare l'evoluzione di una popolazione, rilevare l'emergenza di una soluzione, regolare i parametri dell'algoritmo evolutivo in modo dinamico per evitare comportamenti indesiderati. I titoli degli articoli di Bergmann (et al., 1993) e di Lee (et al., 1993) possono suggerire le parole chiave da cercare nelle pubblicazioni scientifiche.

Rapporti con altri modelli

Neural network

Gli algoritmi genetici possono essere applicati per risolvere problemi come: scegliere la struttura della rete, scegliere i valori dei pesi per ridurre l'errore di output.

Per esempio, un cromosoma è rappresentato dal vettore contenente tutti i pesi dei collegamenti tra gli strati neuronali della rete e ogni singolo peso rappresenta un gene del cromosoma. Ogni individuo viene testato, la funzione fitness coincide con l'esecuzione della rete. I cromosomi migliori sono quelli che portano a un errore globale più vicino al target desiderato. Si parte con un certo numero di cromosomi e si salva il migliore, si crea da esso una nuova popolazione di cromosomi con gli operatori previsti e si prosegue.

Il libro di Iba fornisce conoscenze teoriche e pratiche su una metodologia per la strategia di ricerca evolutiva con l'integrazione di diverse tecniche di machine learning e deep learning, con l'obiettivo di ottenere una migliore ottimizzazione delle metodologie. Viene proposto il modello computazionale denominato Gene Regulatory Network (GRN).

Secondo l'articolo pubblicato su [7] ignorare un obiettivo è il modo migliore per realizzare macchine veramente intelligenti, specialmente quando le neural network prendono in prestito strategie dagli algoritmi evolutivi.