



Algoritmi Genetici

Mai 2018

Ana-Cristina Rogoz

Grupa 232

Facultatea de Matematica și Informatica

Universitatea din Bucuresti

Introducere

Algoritmii genetici pot fi încadrați, alături de cei de tip Monte Carlo, Las Vegas, numerici sau euristici în rândul algoritmilor probabilști. Acest tip de algoritmi a reușit să-și găsească utilitatea întrucât găsi un răspuns suficient de bun în cazul unor probleme ce nu își găsesc soluții polinomiale până în acest moment. Astfel, problemele ce necesită algoritmi probabilști se pot împărți în două categorii: probleme de optim și probleme cu soluție unică. În ambele cazuri restricțiile sunt mai relaxate astfel încât pentru o problema de optim se poate accepta și o soluție suboptimală a carei marja de eroare poate fi controlată cu ajutorul intervalelor de încredere, iar pentru o problemă cu soluție unică poate fi acceptată și o soluție ce se apropie cu o probabilitate mare de soluția exactă.

Algoritmii genetici reprezintă o parte a ceea ce se numește calcul evolutiv. Calculul evolutiv se preocupă în principiu cu algoritmi axați pe găsirea minimului global, iar modul în care aceștia se desfășoară și structura pe care au capătat-o sunt inspirate din biologie, preluând astfel trei mari proprietăți: mutația, selecția și ereditatea. Pentru ca toate acestea să se întâmple, fiecare problema are drept cadru de desfășurare o anumită populație în care fiecare individ este văzut drept soluție, iar principiul predominant pe care se bazează este "supraviețuirea celui mai bine adaptat".

Noțiuni

În cele ce urmează vom face folosință de următoarele noțiuni ce se regăsesc în algoritmi genetici:

- Un **cromozom** reprezintă o mulțime de caracteristici ale unui individ.
- **Populația** reprezintă un ansamblu format dintr-o mulțime de indivizi ce trebuie să se adapteze cât mai bine cadrului în care traiesc.
- **Funcția de fitness/adecvare** ilustrează cât de bine a reușit un individ să se adapteze lumii în care trăiește.
- O **generație** reprezintă o etapă din cadrul evoluției populației.
- **Selecția** este procesul prin care sunt promovați anumiți indivizi în următoarea generație.
- **Operatorii genetici** sunt cele două modificări ce au loc pentru a forma următoarea generație: încrucișarea și mutația.
- **Încrucișarea** are loc între doi cromozomi ce formează indivizi ai următoarei generații și moștenesc caracteristici ale părinților.
- **Mutația** presupune schimbarea aleatoare a uneia sau a mai multor gene ale unui cromozom.

Structura unui algoritm genetic

Această structură este cea propusă de John Holland în anul 1970. Pasul inițial de la care pleacă algoritmul este generarea unei populații inițiale, să spunem de mărime N , ce respectă structura cromozomului din lumea în care lucrăm. Putem considera că lungimea unui cromozom este D , iar fiecare genă a cromozomului poate lua valori dintr-o anumită mulțime - să spunem $C = \{c_1, c_2, \dots, c_k\}$. Vom continua astfel să iterăm prin pașii următori până ce vom atinge condiția de oprire. Având în momentul curent o populație numită să spunem $P(i)$, vrem să generăm indivizii următoarei generații $P(i+1)$.

Începem prin procesul de selecție în care promovăm indivizii ce vor participa ulterior în etapele de încrucișare și mutație. Acest proces de selecție se poate desfășura în mai multe

moduri. Există **selecția proporțională** în care fiecărui individ X din populația curentă $P(i)$ îi asociem o probabilitate $P(X)$ de a fi selectat în funcție de performanța acestuia în raport cu ceilalți după următoarea formulă $f(X) / F$, unde $f(X)$ este funcția de fitness a individului X , iar F este suma tuturor performanțelor indivizilor din populația $P(i)$. De asemenea, mai aveam și **selecția elitistă** care reprezintă trecerea celor mai bine adaptați indivizi în generația următoare, **selecția turneu** în care se aleg aleator câte k indivizi timp de n ori și este promovat din fiecare grup cel cu valoarea funcției de fitness cea mai mare sau selecția **bazată pe ordonare**. Totuși, de cele mai multe ori, selecția pentru care se optează este selecția proporțională ce are la bază metoda ruletei. Această metodă presupune acordarea fiecărui individ un interval în care acesta poate fi selectat. Spre exemplu, putem presupune că avem trei indivizi, primul are probabilitatea 0.2, al doilea are probabilitatea 0.6, iar al treilea are probabilitatea 0.2. Astfel, acordăm fiecăruia un interval de lungimea probabilității sale după cum urmează: $(0, 0.2]$ este dedicat individului 1, $(0.2, 0.8]$ individului 2 și $(0.8, 1]$ individului 3. Încercând un experiment, să spunem că obținem probabilitatea 0.54. Deoarece 0.54 face parte din intervalul individului 2, acesta va fi ales în faza de selecție.

Mai apoi, intrăm în faza de încrucișare unde cromozomii din selecția curentă generează la rândul lor descendenți. Procesul de încrucișare presupune combinarea caracteristicilor a doi indivizi cu scopul formării a doi noi indivizi. De remarcat este însă faptul că nu toți cromozomii ce au reușit să intre în mulțimea de selecție vor participa la încrucișare, ci fiecare dintre ei are o probabilitate de încrucișare fixată, dată încă din parametri de intrare. Încrucișarea poate fi și ea de mai multe feluri: cu un punct de tăietură, cu mai multe puncte de tăietură, uniformă, etc. În urma încrucișării, cei doi descendenți preiau locul "parinților" în populație.

Faza finală pe care algoritmul o desfășoară este mutația. Aceasta presupune schimbarea valorilor unor gene cu rolul de a diversifica cât mai mult populația, dar și de a da șansa unei gene odată dispărute să reapară în ideea că aceasta ar putea reprezenta o parte importantă a soluției finale, chiar dacă există un punct intermediar în care nu mai este prezentă. În funcție de ceea ce ne dorim, mutația poate fi una rară sau dimpotrivă. Prima variantă de mutație, cea rară, presupune că pentru fiecare cromozom în parte să putem


modifica maxim o gena. Parcurgem toți cromozomii din populația curentă și pentru fiecare generăm un număr aleator între 0 și 1, reprezentând probabilitatea de mutație. Acest număr aleator îl comparăm cu probabilitatea de mutație dată de la intrare, iar în cazul în care acesta este mai mare sau egal, atunci vom modifica o genă la întâmplare a cromozomului. De partea cealaltă, opusul mutației rare presupune ca pentru fiecare genă a fiecărui cromozom să generăm un număr aleator pe care să îl comparăm cu probabilitatea de mutație. În acest caz pot exista mult mai multe schimbări de la o generație la alta, iar rezultatele pot deveni neconcludente. Din contă, mutația rară are șanse mai mici să producă astfel de neplăceri, dar aceasta are dezavantajul că poate ajunge mult prea greu la soluția căutată.

În final, am ajuns la formarea unei noi generații de cromozomi. După toate aceste operații, trebuie verificate condițiile de oprire, iar dacă vreuna este îndeplinită algoritmul se sfârșește. Condițiile de oprire de obicei întâlnite sunt fie faptul că am ajuns la numărul maxim de iterații, fie că performanța s-a stabilizat pe parcursul a mai multor generații, fie am ajuns la o soluție destul de bună. Dacă niciuna nu este îndeplinită vom relua pașii anterior amintiți.

Aplicabilități. Exemple

Domenii în care se utilizează frecvent algoritmi genetici sunt robotica, bioinformatica, probleme de trafic, rutare, teoria jocurilor, etc.

Un exemplu potrivit pentru ilustrarea conceptelor algoritmilor genetici este găsirea maximului unei funcții pozitive. Fie $f : [a,b] \rightarrow \mathbb{R}$ o funcție pozitivă, vrem să aflăm X_{\max} , elementul pentru care $f(X_{\max}) \geq f(x_0)$, oricare x_0 din intervalul $[a,b]$. Vom primi de la intrare o serie de date precum intervalul pe care vom lucra $[a,b]$, precizia p reprezentând numărul de zecimale care ne interesează, dimensiunea populației n , numărul de generații t_{\max} , probabilitatea de încrucișare p_c și probabilitatea de mutație p_m . După care avem nevoie de o modalitate de a codifica numerele din intervalul $[a,b]$ astfel încât fiecare cromozom corespunzător unui număr să aibă aceeași dimensiune, iar genele sale să poată



lua aceleași valori. Pentru a putea îndeplini acest aspect, alegem să codificăm binar numerele din intervalul $[a,b]$, dar pentru a respecta precizia vom multiplica toate numerele cu 10^p pentru a ajunge să avem doar numere întregi. Ulterior, aflăm care este lungimea maximă - să o numim l - pe care o poate avea reprezentarea în baza 2 a numerelor din intervalul $[a \cdot 10^p, b \cdot 10^p]$. Având toate acestea calculate putem acum să ne generăm o mulțime de n cromozomi, de lungime l în care fiecare genă poate lua valori în mulțimea $\{0,1\}$. Codificarea fiecărui cromozom poate fi ulterior reprezentată în intervalul $[a,b]$ folosind transformări liniare (prima dată vom duce numerele din baza 2 în baza 10, iar apoi le vom reduce în intervalul $[a,b]$).

Un alt exemplu este găsirea unui anumit string, dându-se lungimea sa. Inițial pornim de la o mulțime de indivizi generată aleator, în care fiecare individ este reprezentat printr-un string, iar mulțimea sa de gene este alcătuită din literele cuvântului ce pot lua valori în mulțimea literelor alfabetului englez. În cazul acesta, funcția de fitness a unui individ reprezintă numărul de caractere din șirul curent ce se identifică cu cele din șirul pe care îl căutăm, iar procesul de încrucișare va consta în alegerea în funcție de o anumită probabilitate fie a genei unui părinte, fie a celuilalt.

Concluzie

Algoritmii genetici reprezintă un nou mod de a analiza diverse probleme ce nu își găsesc soluții polinomiale până în momentul prezent, de obicei din cauza spațiului mare de căutare al soluțiilor. Deoarece aceștia folosesc concepte preluate din biologie, posedă capacitatea de a se adapta de la un pas la altul în funcție de feedback-ul primit datorită funcției de fitness cu scopul de a forma o populație mai bună și mai apropiată de soluția ideală. Astfel, algoritmii genetici pot fi considerați o modalitate interesantă de rezolvare a problemelor, dar mai mult de atât, aceștia aduc cu sine o eficiență ridicată.

Bibliografie

[1] Wikipedia

[2] Algoritmi genetici și strategii evolutive, D. Dumitrescu

[3] Curs "Tehnici avansate de programare", R. Marinescu

[3.1] Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag