Using „machine learning" in Production Rules

Nicolae MĂRGINEAN
Facultatea de Ştiinţe Economice, Universitatea „Babeş-Bolyai" Cluj-Napoca

One of the most expressive and human readable representations is sets of if-then rules. This article explores several algorithms for optimization of production rules. There are presents three way of optimization using genetic algorithms, using decision trees and finally using set of rules.
***Keywords**: decision tree, genetic algorithms, mutation, selection, fitness function, crossover, entropy, information gain, set of rules, overfit*.

1 Reprezentarea cunoaşterii prin reguli de producţie

Modalitatea reprezentării cunoaşterii prin reţele de producţie a fost dezvoltată de Newell şi Simon şi este modalitatea cea mai utilizata în sistemele expert. S-a dovedit foarte eficace pentru reprezentarea recomandărilor, directivelor sau strategiilor atunci când cunoaşterea de reprezentat se referă la experienţa de rezolvare a problemelor dintr-un domeniu particular. Cunoaşterea în reţelele de producţie este de natura procedurală, în care se pot delimita următoarele trei componente:
- cunoaşterea declarativă sau factuală ce reprezintă piese de cunoaştere stocate sub forma unor structuri de date într-o colecţie numită şi context;
- cunoaşterea procedurală reprezentată sub forma unei colecţii de tip condiţie-acţiune numite şi reguli de producţie, colecţie ce formează baza de reguli;
- cunoaşterea strategică sau de control formată din reguli ce privesc secvenţele de acţiuni în procesul de rezolvare.

Sistemul construit în jurul regulilor de producţie se bazează pe o structură specifică compusă din două componente: partea de premisă şi partea de acţiune. Într-o altă exprimare, un sistem de producţie este compus dintr-o bază de fapte şi un set de reguli. Premisele unei reguli pot fi considerate instanţe ale bazei factuale ce returnează un indicator de succes sau eroare. Concluzia unei reguli este o acţiune ce manipulează fapte plus, conform unei strategii de control a sistemului, o secvenţă de alte reguli. Baza factuală este construită dintr-un set de termeni în timp ce regulile au forma generală `IF c THEN t`, în care condiţia c este construită din termeni, paranteze, conective iar concluzia t este formată dintr-un singur termen. Termenii pot fi fie parametrii fie valori.

Reprezentarea exactă a cunoaşterii dintr-un domeniu particular necesită un număr mare de reguli, ca să poată fi surprinse toate detaliile. Numărul regulilor este direct proporţional cu complexitatea expertizei.

Divizarea cunoaşterii în fragmente sau piese de cunoaştere face ca baza de cunoştinţe să fie organizată modular aşa încât poate fi uşor modificată. În plus, metoda regulilor de producţie se bucură de un succes deosebit datorită simplităţii faptelor şi regulilor, formatului liber şi naturaleţei transpunerii cunoaşterii în reguli.

Unul dintre marile dezavantaje ale sistemelor expert clasice bazate pe reguli este incapacitatea acestora de a învăţa. Domeniul "machine learning" nu este altceva decât un răspuns la întrebarea legată de capacitatea programelor de a-şi îmbunătăţii performanţele plecând de la o anumită experienţă. În cazul reţelelor de producţie problema s-ar pune în felul următor: cum s-ar putea ca pe baza unei experienţe anterioare, care în contextul nostru este dată de baza factuală, din regulile existente să putem obţine noi reguli mai performante, mai pertinente.

2. Optimizarea prin algoritmi genetici

O modalitate de îndeplinire a cerinţelor sus amintite ar fi folosirea algoritmilor genetici. Algoritmii genetici oferă o abordare de învăţare bazată pe simularea evoluţiei, sau mai bine spus, pe o analogie cu evoluţia biologi-

că. Aceştia generează ipoteze noi prin aplicarea repetată a unor operaţii de recombinare şi mutaţie asupra celor mai bune ipoteze stabilite la un moment dat. La fiecare pas, o colecţie de ipoteze numită şi populaţia curentă este actualizată prin înlocuiri de segmente ale populaţiei curente. Procesul este o căutare de tip "generează si testează", în care cele mai bune variante generate vor fi luate în considerare pentru generarea unei noi generaţii.

Algoritmii evolutivi reprezintă mai multe clase de metode aleatoare de căutare. Fiecare individ al populaţiilor implicate în procesul de căutare se descrie printr-un singur cromozom.

Orice algoritm evolutiv foloseşte o populaţie de indivizi care este modificată prin intermediul unor operatori genetici cum ar fi cei de selecţie, mutaţie, recombinare,etc. Notăm cu P(t) populaţia de cromozomi la momentul t.

Fie X spaţiul stărilor problemei. Fiecare individ este un element din X. Evaluarea calităţii indivizilor din spaţiul de căutare se face cu ajutorul unei funcţii de performanţă. Fie f:X->R funcţia de performanţă. Fiecare individ este evaluat prin intermediul acestei funcţii. Dacă P(t) este generaţia curentă atunci P(t+1) este noua generaţie obţinută prin selecţia celor mai performanţi indivizi din P(t) şi prin aplicarea asupra lor a operatorilor genetici de recombinare, mutaţie, etc. Operatorul de recombinare este folosit pentru a crea noi indivizi folosind segmente a doi sau mai mulţi cromozomi. Operatorul de mutaţie creează noi indivizi prin schimbări ale unui singur individ.

O variantă de stabilire a funcţiei de adecvare pentru fiecare regulă ar fi aceea prin care funcţia ar fi egală cu procentul cazurilor de aplicare corectă a regulii în numărul total de cazuri, ridicat la pătrat.

Adecvarea(h)=corect(h) * corect(h)

Populaţia iniţială se generează de regulă prin selectarea aleatoare a unor puncte din spaţiul de căutare. De asemenea, algoritmii genetici lucrează cu o codificare a elementelor din spaţiul stărilor problemei şi nu acţionează direct asupra elementelor acestui spaţiu.

Criteriul de oprire pentru un algoritm evolutiv este de regulă legat de numărul de generaţii. Cel mai performant individ al unei generaţii reprezintă soluţia obţinută pentru problema în discuţie. Rezultatul fundamental privind comportarea algoritmilor genetici este teorema schemelor:

O schemă având o adecvare peste medie, valori mici ale ordinului şi lungimii utile, tinde să apară mai frecvent în generaţia următoare. Dimpotrivă, o schemă având adecvarea sub medie tinde să apară mai puţin frecvent în generaţiile următoare.

Fără a intra în amănunte, aceasta teoremă indică faptul că, dacă se vor aplica algoritmii genetici, evoluţia generaţiilor viitoare are loc în sensul creşterii performanţei. Altfel spus, aplicând operatorii genetici (mutaţie, recombinare etc.) asupra unor blocuri constructive se obţin noi blocuri constructive tot mai performante, care vor converge spre soluţia optimă.

Să urmărim în continuare o modalitate de îmbunătăţire a regulilor de producţie apelând la algoritmii genetici.

Încă din start trebuie precizaţi cei doi parametrii ai algoritmului, probabilitatea de încrucişare pc şi probabilitatea de mutaţie pm.

Prima operaţiune ce trebuie efectuată este codificarea elementelor ce reprezintă populaţia iniţială. În cazul nostru populaţia iniţială este dată de un set de reguli. Se ia fiecare regulă în parte şi pentru fiecare termen al regulilor, se determină toate variantele posibile. După aceea folosind o codificare binară se formează un şir de 0 si 1 de lungime dată de numărul de variante în care pentru fiecare variantă existentă într-o anumită regulă se atribuie valoarea 1 si pentru fiecare varianta inexistentă în regulă de codificat se pune 0.

Ex. `Dacă vremea(ploioasa OR rece) AND vântul(puternic) atunci activitate(învăţ)`

```
Vremea – poate fi rece, ploioasă, caldă
110
Vântul – puternic, slab 10
Activitate – învăţ, dorm 10
```

Codificarea integrală se obţine prin concatenarea codificărilor aferente fiecărui termen.

Ex: `1101010`

La fel se procedează pentru toate regulile.

După ce am terminat de codificat populaţia iniţială se va aplica următorul algoritm iterative:

1. se iniţializează populaţia P(t)
2. se evaluează fiecare element al populaţiei P(t)
3. se aplică operatorul de selecţie de n ori. Indivizii selectaţi formează o populaţie intermediară în care se reţin cei mai performanţi indivizi. De reţinut că un individ poate să apară de mai multe ori.
4. Se aplică încrucişarea; se şterg din populaţie părinţii indivizilor obţinuţi prin încrucişare
5. Se aplică mutaţia
6. t=t+1
7. Dacă t<=n, unde n este numărul maxim de generaţii, atunci se revine la pasul 2
8. Altfel se decodifică şirurile nou obţinute rezultând noi reguli mai performante

În continuare să detaliem cele trei operaţii de selecţie, încrucişare şi mutaţie.

Selecţia

Există mai mulţi algoritmi de selecţie dar cel mai uzual este cel al selecţiei proporţionale. În acest caz, selecţia unui individ depinde de performanţa acelui individ. Se va utiliza o metoda de tip Monte Carlo.

1. se calculează funcţia de performanţă pentru fiecare individ
2. se calculează suma funcţiilor de adecvare a fiecărui individ rezultând performanţa totală a populaţiei curente $F = \sum_{i=1}^{n} f(x_i)$.
3. probabilitatea de selecţie a individului i va fi $p_i = \frac{f(x_i)}{F}$.
4. pentru fiecare individ se calculează $q_i = \sum_{k=1}^{i} p_k$.
5. pentru fiecare individ se generează un număr aleator g în intervalul [0,1]
6. dacă $0 \le g \le q_1$ se selectează primul individ; dacă $q_{i-1} < g \le q_i$ se selectează individul i.

Încrucişarea

1. pentru fiecare individ se generează un număr aleator q în [0,1].
2. dacă q<pc, unde pc este probabilitatea de încrucişare, individul este reţinut pentru încrucişare, altfel este neglijat
3. după ce am reţinut m indivizi aleşi mai sus, se formează [m/2] perechi în mod aleator
4. pentru fiecare pereche se generează un număr aleator întreg între 1 şi r, lungimea indivizilor. K obţinut mai sus reprezintă punctul de tăietură, după care se efectuează încrucişarea propriu zisă, adică segmentele 1..k de la cei doi indivizi se schimbă între ele la fel cum segmentele k+1..r ale celor doi indivizi se schimbă între ele.
5. Indivizii nou obţinuţi devin membrii ai unei noi populaţii în timp ce părinţii acestora se vor şterge

Mutaţia

1. se generează un număr aleator q între 0 şi 1 pentru fiecare poziţie dintr-un individ
2. dacă q<pm, unde pm este probabilitatea de mutaţie, se execută o mutaţie, adică se schimbă 0 în 1 şi 1 în 0; în caz contrar poziţia nu se schimbă.

3. Optimizarea prin arbori decizionali

Arborele decizional este unul din mijloacele cele mai folosite în practică pentru inferenţa inductivă. El are în structura sa noduri ce reprezintă scopurile iar arcele reprezintă valorile posibile ale nodurilor numite şi acţiuni sau decizii. Acesta se citeşte de la stânga la dreapta şi de sus în jos. De obicei rădăcina se află în partea de sus. Toate nodurile cu excepţia rădăcinii sunt instanţe ale nodului principal. Avantajul din punctul de vedere al studiului de faţă este acela că arborii decizionali pot fi reprezentaţi sub forma regulilor de producţie. Un arborele decizional nu este altceva decât o disjuncţie de conjuncţii sau reguli de producţie. Calea de la rădăcina arborelui până la o frunză formează o regulă de producţie (un ansamblu de conjuncţii ce va implica ceva) unde nodurile parcurse formează partea de premisă iar frunza arborelui reprezintă partea de acţiune. Iar dacă mergem invers, şi ceea ce ne interesează pe noi, o regulă de producţie dată într-o manieră nerecursivă poate fi transformată într-o ramură a unui arbore decizional.

Scopul nostru este ca plecând de la un mănunchi de reguli, să construim un arbore decizional optim pe baza căruia prin transformarea amintită mai sus să obţinem un nou set

de reguli mai complet, mai bun.
Unul din algoritmii folosiţi într-un astfel de context ar fi algoritmul cunoscut în literatura de specialitate "algoritmul ID3".
Problema majoră care se pune este aceea de stabilire a ordinii în care scopurile reprezentate prin noduri vor fi aranjate într-un arbore aşa încât să obţinem organizare optimă. Construirea arborelui se va face într-o maniera top-down, începându-se cu rădăcina şi terminând cu frunzele arborelui. Pentru fiecare nod se va stabili o metrică statistică ce va permite ierarhizarea acestor noduri. Aceasta metrică poarta numele de indicatorul câştigului de informaţie. Acesta este într-o strânsă legătură cu un alt indicator, entropia informaţională, des întâlnită în teoria informaţională. Entropia informaţională caracterizează gradul de dezordine dintr-o colecţie arbitrară de exemple.
Astfel, dându-se o colecţie S, conţinând exemple negative şi pozitive, entropia acestei colecţii va fi dată de:

$$Entropia(S) = -p_1 \log_2 p_1 - p_2 \log_2 p_2$$

unde P1 este procentul exemplelor pozitive în numărul total de exemple; P2 este procentul exemplelor negative în numărul total de exemple
Se observă că entropia este maximă, adică 1, în momentul în care numărul exemplelor negative este identic cu numărul exemplelor pozitive. Entropia este minimă, adică 0, în momentul în care avem doar elemente pozitive sau doar elemente negative.
Revenind la indicatorul câştigului de informaţie al unui nod, acesta ne oferă o măsură a reducerii entropiei prin partiţionarea exemplelor prin acel nod

$$castig(S, A) = entropia(S) - \sum_{v \in Valori(A)} \frac{|S_v|}{|S|} Entropia(S_v)$$

unde S – colecţia de exemple, |S| - numărul de elemente al colecţiei S, A-nodul de analizat, Valori(A) - valorile posibile ale nodului A). Astfel, calculăm indicatorul câştig de informaţie pentru fiecare nod determinând care dintre ele are cea mai mare valoare a indicatorului. Nodul astfel obţinut va fi plasat ca rădăcina a arborelui.
Pentru fiecare valoare posibilă a nodului determinat mai sus, nodurilor rămase li se calculează acelaşi indicator ţinând seama că de această dată avem ca şi colecţie de pornire doar acele elemente din mulţimea iniţială ce au în comun valoarea amintită mai sus: S→ Sv. În orice moment, nodul cu câştig de informaţie mai mare va fi plasat în arbore, formând o nouă subramură. Algoritmul continuă până când toate frunzele arborelui au entropie zero. După ce algoritmul ID3 se încheie, se transformă arborele nou obţinut în reguli de producţie.
La arborii decizionali poate apărea următoarea problemă. Este posibil ca un arbore decizional, cu cât este mai stufos să reprezinte mai bine setul de exemple de la care a pornit dar cu atât mai mult să nu se conformeze unor noi seturi de exemple. Este demonstrat faptul că începând cu dimensiuni de peste 25 de noduri, arborii reprezintă din ce în ce mai prost noile seturi de exemple. De aceea se preferă arborescenţele mai mici de 25 de noduri. Pentru a preîntâmpina acest neajuns, pentru arbori de dimensiuni de peste 30 de noduri, se foloseşte următorul algoritm al lui Quinlan, C4.5.
Arborele decizional obţinut prin ID3 se transformă în reguli de producţie prin metoda cunoscută (calea *rădăcină → frunză*).
Pentru fiecare regulă obţinută, se elimină pe rând câte o precondiţie din partea de premisă a regulii, începând cu cele din partea de final a premisei regulii. Dacă în urma calculului unui indicator de adecvare pentru noua subregulă se obţin valori mai mari decât cele pentru acelaşi indicator de adecvare al regulii iniţiale, se elimină regula iniţială şi se păstrează subregula obţinută. După cum se observă, se ţine seama de următoarea afirmaţie demonstrată statistic. Cu cât indicatorul de adecvare este mai mare, cu atât va clasifica mai prost noile seturi de exemple. De aceea se apelează la eliminarea de noduri.
Gradul de adecvare de care aminteam se defineşte prin următorul interval:

$$eroare(h) \pm z_n \sqrt{\frac{eroare(h)(1 - eroare(h))}{n}}.$$

unde Zn se ia dintr-un tabel (pentru un grad de încredere de 95%, Zn=1.96)
Eroare(h) este procentul cazurilor eronate în numărul total de exemple *n*.

4. Optimizarea prin generarea unor noi seturi de reguli.

Până în acest punct, am analizat două metode de optimizare a regulilor de producție. Prima, folosind algoritmii genetici unde regulile erau reprezentate sub forma unor şiruri de biți iar a doua cea în care în prealabil se generau arbori decizionali şi pe baza acestora se descriau regulile, parcurgând arborii de la rădăcină la frunză. După cum se observă, ambele au folosit o anumită codificare a regulilor. De această dată, vom lucra direct cu reguli..

Vom analiza pentru început regulile propoziționale, adică acelea care nu conțin variabile. Pentru aceasta vom folosi *algoritmii de acoperire secvențială*. Aceştia presupun că se determină o regulă care atinge un anumit grad de acuratețe, se şterg toate exemplele pozitive acoperite de aceasta regulă şi se trece la determinarea unei noi reguli pentru exemplele pozitive rămase. Ca şi grad de acuratețe se va folosi fie entropia(cu cât entropia este mai joasă cu atât gradul de performanță este mai mare) fie frecvența relativă (nc/n) unde nc este numărul de potriviri iar n este numărul de exemple. În generarea unei reguli se pleacă de la regula fără premisă. Treptat, prin generarea tuturor variantelor posibile, se adaugă în premisă precondiții într-o manieră greedy(se obține un optim local, nu neapărat global) până când se atinge gradul de performanță dorit. Pentru a preîntâmpina acest aspect negativ, se poate adopta şi strategia în care se memorează cele mai bune k variante pentru fiecare adăugare de precondiție.

În ceea ce priveşte algoritmul de aplicare în cazul regulilor predicative de ordinul întâi, acesta diferă față de cel prezentat mai sus din două perspective: al modalității de generare a regulii şi al indicatorului de performanță. În ceea ce priveşte generarea, literalii care se adaugă în premisa regulii trebuie să aibă următoarea formă:

1. q(v1,…vr) unde q este unul din predicatele disponibile iar vi poate fi o variabilă nouă sau una existentă. Cel puțin una dintre variabile trebuie să se regăsească în vechea regulă;
2. predicatul de egalitate între două variabile existente deja într-o regulă;
3. negarea formelor de mai sus;

Ca grad de performanță se va folosi următorul indicator:

$$castig\,(L,R) = t(\log_2 \frac{p1}{p1+n1} - \log_2 \frac{p0}{p0+n0})$$

Unde p0 este numărul de potriviri pozitive ale regulii R, n0 este numărul de potriviri negative ale lui R, p1 este numărul de potriviri pozitive ale noii reguli L+R, n1 este numărul de potriviri negative ale noii reguli, R regula la care se adaugă un literal, L literalul care se adaugă, t numărul de potriviri pozitive înainte şi după adăugarea literalului L la R.

Concluzii

În tot ceea ce am analizat în studiul de față nu am făcut nimic altceva decât să încercăm să înzestrăm sistemele bazate pe reguli de producție cu capacitatea de a învăța, înlăturând astfel neajunsul amintit la începutul acestui studiu. Variantele metodelor prezentate mai sus reprezintă doar o părticică din tot ceea ce înseamnă domeniul "machine learning". Sistemele expert, ca şi componentă de bază a sistemelor suport de decizie inteligente, îşi sporesc astfel abilitățile; pe lângă dispunerea de expertiză, raționament simbolic, profunzime şi autocunoaştere, am mai adăugat o nouă abilitate, aceea de a-şi optimiza regulile şi-n acelaşi timp de a învăța.

Bibliografie

1. Tom M Mitchel, *Machine Learning*, McGraw-Hill Companies, 1997
2. Ioan Andone, Alexandru Țugui, *Dezvoltarea sistemelor inteligente în economie*, Editura Economică, Bucureşti, 2001
3. Efraim Turban, *Decision Support System and Expert System*, McMillan Publishing Company, New York, 1993
4. D. Dumitrescu, *Algoritmi genetici şi strategii evolutive* – aplicații în Inteligența artificială şi în domenii conexe, Editura Albastră, Cluj-Napoca, 2000
5. D. Dumitrescu, *Principiile inteligenței artificiale*, Editura Albastră, Cluj-Napoca, 1999
6. Brodley,CE & Utgoff, *Machine Learning*, 1995