Paralelizarea algoritmilor evolutivi pentru instruirea maşinilor cu vectori suport

Gheorghe Radu, Ştefan-Gheorghe Pentiuc, Ionuț Bălan

***Abstract*—Evolutionary Approach to Support Vector Machines (EASVMs) is a novel technique constructed as an alternative of the standard SVMs architecture (R. Stoean et al, 2009). In this paper we present new parallel algorithms for EASVMs. The experimentation were deployed on a cluster of High Performance Computing Laboratory in "Stefan cel Mare" University of Suceava. The computational results show the validity of the approach in terms of runtime, accuracy and speedup.**

***Index Terms*—Evolutionary algorithms, parallel computing, supervised learning, support vector machines.**

I. INTRODUCERE

SVM-urile (maşinile cu vectori suport) s-au dovedit a fi intrumente deosebit de puternice pentru clasificare şi regresie. Totuşi, in ciuda originalităţii şi performanţei viziunii de învăţare, motorul de instruire apare ca fiind restrictiv, rareori transparent, capabil de a converge numai pentru câteva funcţii de decizie particulare. Această situaţie a motivat cercetătorii în a investiga modalităţi alternative de instruire a SVM. În acest articol vom prezenta o abordare evolutivă a SVM-urilor, EASVM – *evolutionary approach to support vector machines*, urmărind, în principal, lucrarea [1], cu menţiunea că primele idei legate de o astfel de abordare au fost cuprinse în [2].

Algoritmii evolutivi sunt algoritmi robuşti, dar care necesită un volum apreciabil de resurse pe parcursul rulării lor. O metodă de paralelizare a algoritmilor evolutivi (EA) pentru instruirea SVM, precum şi rezultatele obţinute, vor fi deasemenea prezentate în această secţiune, urmărind [3].

Optimizarea evolutivă permite adaptarea hiperplanului de decizie la datele de instruire disponibile, rezolvând direct problema de optimizare primală. Este luată în considerare ideea de bază a SVM, conceptul geometric de instruire, dar abordarea propusă se abate de la tratarea matematică standard. În plus, tehnicile evolutive deschid calea generalizărilor ce implică suprafeţe de decizie nestandard, neliniare. Se propune astfel o nouă abordare: instruirea urmăreşte SVM-urile standard, în timp ce valorile optime ale coeficienţilor hiperplanului (w şi b) sunt determinate direct de un algoritm evolutiv, cu respectarea echilibrului între acurateţea clasificării şi abilitatea de generalizare (vezi şi [4]). Deşi reprezentarea sugerată pare să fie simplă, determinarea altor detalii ale algoritmului (cum ar fi: interpretarea, operatorii, parametrii) nu este aşa ! În abordarea pe care o prezentăm, calculul evolutiv al coeficienţilor funcţiei de decizie se face în cadrul conceptului geometric de instruire a SVM.

II. ALGORITMUL EASVM (ALGORITM EVOLUTIV PENTRU INSTRUIREA SVM)

Problema primală generală de găsire a hiperplanului de decizie (vezi şi [5], pag. 67) este rezolvată utilizand cu consecvenţă un algoritm evolutiv. Elementele evolutive ale acestui algoritm sunt prezentate în continuare:

Reprezentarea: Coeficienţii hiperplanului, w si b, sunt codificaţi în structura unui cromozom: $c=(w_i,\ldots,w_n,b)$. Initial, cromozomii sunt generaţi aleator, astfel încât

$$w_i \in [-1,1],\ i=1,2,\ldots n,\ b \in [-1,1]. \quad (1)$$

Asignarea (funcţiei de) fitness: Funcţia de fitness derivă din funcţia obiectiv şi este supusă restricţiilor problemei de optimizare. Depărtându-se, însă, de SVM standard, pentru funcţia de fitness este derivată o formulare neliniară, diferită. Parametrul w este mapat din Φ în H. Rezultă, deci, că norma pătratică implicată în condiţia de generalizare devine $\left\|\Phi(w)\right\|^2$, iar ecuaţia hiperplanului de separare este $\left\langle \Phi(w), \Phi(x_i) \right\rangle - b = 0$.

Este utilizată forma $\left\langle u, w \right\rangle = u^{\mathrm{T}} w$ şi nucleul este folosit pentru a transforma norma într-un produs scalar. Formularea condiţiei (funcţiei) de fitness (de a fi minimizată) încorporează funcţia obiectiv şi restricţiile sunt formulate prin penalizarea cromozomilor eronat clasificaţi printr-o funcţie t care returnează valoarea argumentului, dacă acesta e negativ şi valoarea 0, în caz contrar. Pentru clasificare, expresia funcţiei de fitness este :

$$f(w,b)=\frac{1}{2}K(w,w)+C\sum_{i=1}^{m}\xi_i+$$

Gheorghe RADU is with "Henri Coandă" Air Force Academy, #160, M. Viteazul street, 500183 Braşov, gh.radu@gmail.com. He is also PhD student at "Ştefan cel Mare" University of Suceava

Ştefan-Gheorghe PENTIUC, PhD, is with "Ştefan cel Mare" University of Suceava, #13, Universităţii street, 720229 Suceava, pentiuc@eed.usv.ro

Ionuț BĂLAN is with "Ştefan cel Mare" University of Suceava, #13, Universităţii street, 720229 Suceava, ionutz19sv@yahoo.com. He is also PhD student at "Ştefan cel Mare" University of Suceava

$$\sum_{i=1}^{m}[t(y_i(K(w,x_i)-b)-1+\xi_i)]^2 \quad (2)$$

Selecţia şi operatorii de variaţie : Sunt aplicate cele mai uzuale scheme folosite în codificarea reală. Acestea sunt: selecţia turnir, încrucişarea intermediată şi mutaţia cu perturbare (distribuţie) normală.

Condiţia de oprire : Algoritmul se opreşte după un număr predefinit de generaţii.

Îndată ce sunt găsite cele mai apropiate valori optimale ale coeficienţilor hiperplanului de decizie, o nouă formă poate fi clasificată direct, prin utilizarea ecuaţiei:

$$class(x_i)=\mathrm{sgn}(K(w,x_i)-b). \quad (3)$$

Acurateţea clasificării este definită ca fiind raportul dintre numărul cazurilor/formelor corect etichetate şi numărul total de exemple de test.

A. *O construcţie naivă a EASVM*

Vrem să evaluăm dacă algoritmul EASVM poate da rezultate bune atunci când este comparat cu abordarea standard a SVM. Mai întâi, trebuie să determinăm parametrii potriviţi ai algoritmului. Astfel, dacă indicatorii de erori $\xi_i,\ i=1,2,...,m$ apar în restricţiile formulate pentru optimalitatea planului, aceştia trebuie trataţi (corespunzător) şi în algoritmul EASVM. În acest caz, algoritmul EASVM se depărtează de înţelesul strict geometric (al unei SVM canonice) al unei deviaţii şi, pur şi simplu, evoluează indicatorii de erori. Prin urmare, algoritmul EASVM include aceşti indicatori în structura unui cromozom, codificarea acestuia devenind :

$$c=(w_1,...,w_n,b,\xi_1,...,\xi_m), \quad (4)$$

unde $w_i\in[-1,1],\ i=1,2,...,n,\ b\in[-1,1]$ şi $\xi_j\in[0,1],\ j=1,2,...,m$.

Construcţia naivă a EASVM produce, de asemenea, bune rezultate şi când se compară cu SVM canonic. În orice caz, abaterile standard mai mici dovedesc o stabilitate mai înaltă a abordării EASVM. Trebuie de asemenea sa remarcăm că, pentru nuclee standard, nu ne putem aştepta ca algoritmul EASVM să fie considerabil mai bun decât SVM standard, din moment ce transformarea nucleu ce induce instruirea este aceeaşi. În orice caz, flexibilitatea algoritmilor evolutivi ca instrumente de optimizare face din algoritmul EASVM o alegere atractivă din perspectiva performanţei, datorită abilităţii prospective a acestuia de a evolua (în plus) nuclee adaptate la problemă, indiferent dacă aceste nuclee sunt pozitiv (semi-)definite sau nu, ceea ce este imposibil pentru SVM canonice.

Totuşi, pentru mulţimi mari de date (spam filtering, de exemplu) timpul de execuţie necesar instruirii este foarte mare. Aceasta rezultă din faptul că se folosesc indivizi/cromozomi foarte mari, cu foarte multe caracteristici (gene), cum ar fi indicatorii de eroare ai tuturor formelor din mulţimea de instruire, care sunt incluşi în reprezentare. Această problemă poate fi rezolvată în mai multe moduri :

- prin aplicarea procedurii chunking ([Perez04]), timpul de executie se reduce de cca opt ori, cu preţul unei mici pierderi în ceea ce priveşte acurateţea; în afară de rezolvarea problemei lungimii unui cromozom, mecanismul propus reduce numărul mare de calcule necesare referirii formelor de instruire în expresia funcţiei de fitness;
- prin paralelizarea algoritmului EASVM; această metodă va fi prezentată în capitolul următor.

B. *Algoritmul EASVM simplificat*

Chiar dacă algoritmul EASVM ne oferă o alternativă viabilă la SVM, el poate fi încă îmbunătăţit în ceea ce priveşte simplitatea. Problema curentă de optimizare necesită tratarea indicatorilor de erori care, în prezenta variantă de algoritm evolutiv, sunt incluşi în reprezentare. Aceasta poate complica sever problema prin creşterea lungimii cromozomului (numărul de variabile) cu numărul de forme de instruire. Mai mult decât atât, o astfel de metodologie se depărtează puternic de conceptul de SVM canonic. În această secţiune este prezentată o procedură de reprezentare numai a coeficienţilor hiperplanului şi de calcul al indicatorilor de erori (în loc de a-i evolua), procedură care îşi menţine performanţa.

Din moment ce algoritmul EASVM furnizează direct şi interactiv coeficienţii hiperplanului în orice moment, se propune scoaterea indicatorilor de eroare din reprezentarea algoritmului evolutiv şi, în schimb, calcularea acestor valori. În consecinţă, reprezentarea unui individ conţine numai *w* şi *b*. În plus, toţi indicatorii $\xi_i,\ i=1,2,...,m,$ vor putea fi calculaţi în vederea referirii lor în funcţia de fitness. În ceea ce priveşte clasificarea, se ia individul curent (hiperplanul de separare), iar hiperplanele suport sunt determinate prin mecanismul propus de Bosch şi Smith, în lucrarea [Bosch98]. Mai întâi calculăm :

$$\begin{aligned} &m_1=\min\{K(w,x_i)\,|\,y_i=+1\}si\\ &m_2=\max\{K(w,x_i)\,|\,y_i=-1\},\quad i=1,2,...,m;\\ &p=|m_1-m_2|;\\ &w'=\left(\frac{2}{p}\right)w;\ b'=\left(\frac{1}{p}\right)(m_1+m_2). \end{aligned} \quad (5)$$

Pentru fiecare formă de instruire $x_i,\ i=1,2,...,m$, deviaţia faţă de hiperplanul suport corespunzător este obţinută astfel :

$$\begin{aligned} &\delta(x_i)=k(w',x_i)-b'-1,\quad y_i=+1;\\ &\delta(x_i)=k(w',x_i)-b'+1,\quad y_i=-1. \end{aligned} \quad (6)$$

Dacă semnul deviației unei forme este egal cu clasa, atunci indicatorul său este 0; altfel, deviația absolută (normalizată) este returnată ca indicator de eroare. Forma funcției de fitness rămâne ca în formula (2), evident fără a lua ξ_i ca argumente ale funcției.

III. PARALELIZAREA ALGORITMULUI EASVM

Prin utilizarea algoritmilor evolutivi în rezolvarea unei probleme, rezultatele sunt obținute într-un timp destul de mare, cu un consum important de resurse. Datorită faptului că acest tip de algoritmi pot fi relativ uşor paralelizați, am recurs la această variantă, în primul rând cu scopul de a obține rezultatele din ce în ce mai bune, într-un timp rezonabil de mic.

A. *Structura paralelă folosită*

Testele au fost efectuate pe un cluster (RedPower) cu procesoare XEON Quad Core, din dotarea laboratorului de Calcul de Înaltă Performanță al Universității „Ştefan cel Mare" din Suceava. Această arhitectură este formată din 28 de noduri, fiecare nod având 8 nuclee, viteza ceasului fiind de 2,3 GHz, iar memoria pe nod de 1,96 GB.

Comunicațiile dintre nodurile folosite în testele pe care le-am efectuat se realizează folosind biblioteca OpenMPI [6]. Paralelizarea acestui algoritm a fost realizată la nivel de date, în sensul că populația este împărțită în mod egal între procesele MPI ce rulează pe fiecare nod al clusterului, alegându-se un model insular ce permite acest lucru.

B. *Datele de test*

Pentru testarea algoritmului propus s-au folosit patru seturi de date [7]:

-Pima Indians Diabetes Data Set;
-Iris Data Set;
-Spambase Data Set;
-Soybean (Small) Data Set.

În fiecare caz considerat s-au făcut câte 5 rulări, luând-se în considerare valorile medii ale rezultatelor obținute. În Tabelul 1 sunt prezentate principalele caracteristici ale seturilor de date, aşa cum apar în cadrul algoritmului genetic în fiecare caz în parte.

TABEL 1 CARACTERISTICILE SETURILOR DE DATE.

Baza de date	Kernel	Numărul formelor de antrenare	Număr formelor de testare
Pima	polinomial	576	192
Iris	radial	105	45
Spambase	polinomial	3451	1150
Soybean	polinomial	30	17

Trebuie să precizăm că dimensiunea populației în fiecare caz în parte este de 200 indivizi, numărul de generații în care vor evolua aceste populații este de 250, în timp ce atât probabilitatea de recombinare, cât şi cea de mutație este de 0,4.

IV. REZULTATELE EXPERIMENTELOR

Rezultatele obținute în urma rulărilor pe diferite arhitecturi sunt prezentate în Tabelul 2.

TABEL 2 ACURATEȚEA MEDIE OBȚINUTĂ.

Numărul de noduri	Iris(%)	Spam(%)	Pima(%)	Soybean(%)
1	91,99	74,83	72,93	90,72
2	94,66	77,15	77,83	92,71
4	97,32	78,13	78,32	97,13
8	98,11	79,13	79,15	100
20	99,97	79,45	79,23	100

Datele din tabelul 2 sunt reprezentate graphic în Figura 1, pentru a scoate în evidență câstigurile obținute prin paralelizarea algoritmului evolutiv din cadrul problemei rezolvate.

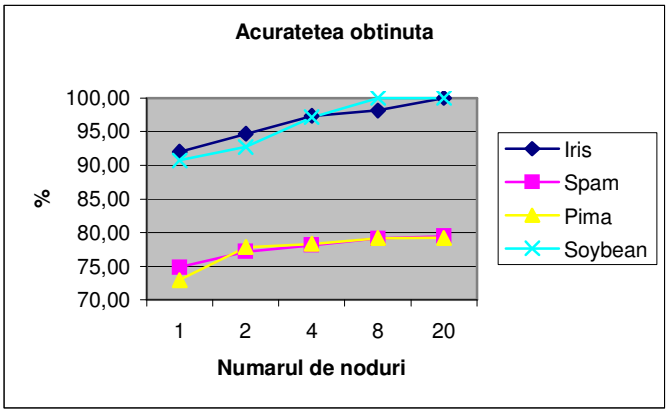

Fig.1 Acuratețea obținută în cele 4 cazuri.

Pe seturile de date studiate se observă că introducerea paralelizării algoritmilor genetici a condus la îmbunătățirea rezultatelor în marea majoritate a cazurilor avute în vedere. În cazul bazei de date Soybean se observă că la 20, respectiv 8 noduri folosite pentru rulare se atinge acuratețea de 100%. Acest rezultat apare ca urmare a numărului mic de forme ce trebuie incluse înr-o anumită categorie, aceasta realizându-se cu probabilitate mult mai mare decât în cazul seturilor de date (mult) mai mari.

Un anumit cluster dispune de resurse suficiente pentru rezolvarea anumitor probleme. Prin urmare, nu are rost să supraîncărcăm cu sarcini anumite procesoare, doar pentru obținerea unor rezultate mai bune, în schimbul creşterii duratei de rulare. Pentru a confirma acest lucru am recurs la utilizarea aceluiaşi număr de indivizi într-o anumită generație de evoluție, dar pe configurații diferite. Rezultatele obținute prin rularea algoritmului asupra setului de date Iris, în condițiile precizate mai sus, sunt prezentate în Tabelul 3.

TABEL 3 REZULTATE OBȚINUTE PRIN RULAREA, ÎN DIFERITE CONFIGURAȚII, A SETULUI DE DATE IRIS.

Dimensiunea populației	**4000**	**2000**	**800**	**400**	**200**
Numărul de procesoare	1	2	5	10	20
Timpul (s)	129,3	63,93	26,33	12,86	6,69
Acuratețea(%)	92,33	100	98,33	97,77	100

Din Tabelul 3 putem trage concluzia că nu este necesar să folosim populații imense, ce pot duce la sufocarea unei anumite unități de calcul în momentul în care avem la dispoziție mai multe unități de procesare ce sunt legate în paralel. Prin recurgerea la astfel de metode, folosind același număr de indivizi/generație, obținem rezultate mult mai bune și timpi de rulare semnificativ mai mici.

Pentru a putea la răspunde la întrebarea: „este benefică paralelizarea algoritmilor genetici pentru rezolvarea problemelor propuse ?”, vom calcula accelerarea obținută pe baza datelor din Iris Data Set. Valorile obținute sunt reprezentate în Figura 2.

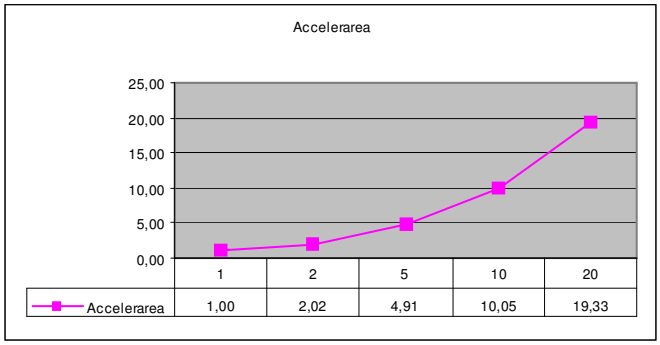

Fig. 2 Accelearea obținută prin paralelizarea algoritmilor genetici.

Teoretic, accelerarea nu poate depăși numărul de procesoare pe care s-a realizat paralelizarea, dar este de dorit și de așteptat să se apropie cât mai mult de acesta. În cazul nostru se observă anumite diferențe între accelerarea obținută și numărul de procesoare pe care s-a făcut rularea. Principala cauză care a dus la această situație fiind existența unor porțiuni mari de program (diferite de cele rezervate algoritmilor genetici) ce nu au fost (sau nu au putut fi) paralelizate. O altă cauză, minoră, pentru această diferență ar putea fi timpii necesari comunicațiilor dintre procesoare.

V. CONCLUZII

Abordarea EASVM materializează viziunea de învățare a SVM, dar are un mare avantaj în ceea ce privește transparența procesului de instruire, inerent oricărei probleme de optimizare prin intermediul unui algoritm evolutiv.

Sunt utilizate două reprezentări ale algoritmului evolutiv: una este simpla, dar rapidă, cealaltă este mai complicată, dar dovedește o acuratețe mai bună.

Paralelizarea algoritmului EASVM se dovedește a fi o cale extrem de promițătoare, atât în ceea ce privește performanța demonstrată în rezolvarea problemelor (foarte) mari (chiar și fără să fie utilizat mecanismul de *chunking*), cât și acuratețea soluțiilor, superioară în toate cazurile.

REFERINȚE

[1] R. Stoean, M. Preuss, C. Stoean, E. El-Darzi, D. Dumitrescu, *Support vector machines learning with an evolutionary engine*, Journal of the Operational Research Society **60**, doi:10.1057/jors.2008.124, pp. 1116-1122, 2009.

[2] Gh. Radu**,** *Tehnici evolutive de instruire a SVM,* Universitaria Ropet 2003: lucrările Simpozionului Internațional, vol. Matematică-Informatică și Fizică, pp. 97-102, Editura UNIVERSITAS, Petroșani, 2003 (ISBN 973-8260-37-X).

[3] Gh. Radu**,** I. Bălan, Șt.-Gh. Pentiuc, *Learning Support Vector Machine Using Parallelized Evolutionary Algorithms*, in Proceedings of the International Conference on Theory and Applications of Mathematics an Informatics, ICTAMI 2011, Alba Iulia, special issue of ACTA UNIVERSITATIS APULENSIS, Mathematics-Informatics, 2011, pp. 331-340, ISSN 1582-5329.

[4] R. Stoean, M. Preuss, C. Stoean, D. Dumitrescu, *Evolutionary Support Vector Machines and their Application for Classification,* Technical Report Nr. CI-212/06, Collaborative Research Center on Computational Intelligence, University of Dortmund, June, 2006.

[5] Gh. Radu**,** Șt.-Gh. Pentiuc, *Mașini cu vectori suport nuanțate – o nouă abordare în clasificarea datelor,* Sisteme distribuite ISSN 1842-6808, pp. 65-70, 10 Dec 2010, Suceava, http://eed.usv.ro/SistemeDistribuite/2010/Masini%20cu%20vectori.pdf.

[6] G. Burns, R. Daoud, J. Vaigl, *LAM: An Open Cluster Environment for MPI,* in Proceedings of Supercomputing Symposium, pp. 379-386, 1994.

[7] A. Frank, A. Asuncion, *UCI Machine Learning Repository*, University of California, Irvine, School of Information and Computer Science, http://archive.ics.uci.edu/ml, 2010.