

**UNIVERSITATEA "LUCIAN BLAGA" DIN SIBIU**

**Contract UEFISCDI nr. 27 / 04.08.2010**

**Programul: Resurse umane**

**Tipul proiectului: Proiect de cercetare post-doctorală**

**Cod proiect: PD\_670 / 28.07.2010**

**SINTEZA DE CERCETARE**

**SISTEM DE CLASIFICARE AUTOMATA A DATELOR NESTRUCTURATE  
FOLOSIND METACLASIFICATOARE BAZATE PE METODE DE TIP SUPPORT  
VECTOR MACHINE ȘI NAIVE BAYES**

**Etapa unica 2011  
Anul II**

**DIRECTOR PROIECT,**

**Sef lucrari dr. Ing. Daniel MORARIU**

**decembrie, 2011**

# 1 Rezumatul proiectului

Acest proiect se încadrează în domeniul clasificării automate a documentelor text și își propune îmbunătățirea rezultatelor clasificării prin realizare unor strategii de combinare a rezultatelor metodelor de clasificare folosite.

Un număr impresionant de documente se găsesc în format electronic, iar utilizatorul are nevoie de componente de clasificare automată a acestora pentru a le putea gestiona. Gestiunea lor a devenit o problema foarte importantă în ultima perioadă. Devine esențială existența unor programe inteligente de organizare automată a documentelor în categorii pentru a facilita analiza și prelucrarea acestor documente. Datorită domeniului foarte vast în care ar trebui să lucreze acestea, devine dificil de realizat un singur clasificator cu performanțe foarte bune. Abordarea actuală este de a utiliza mai mulți clasificatori de diferite tipuri combinați într-un meta-clasificator, sau realizarea unei clasificări hibride care se bazează pe predicția clasificatorului cel mai bun pentru o problema particulară folosind caracteristicile vectorilor de intrare ale documentelor și istoria clasificărilor. Având mai mulți clasificatori de bază, de tipuri diferite (SVM - Support Vector Machine, Bayes, rețele neurale, etc), ideea este de a învăța un meta-clasificator care prezice gradul de corectitudine pentru fiecare dintre clasificatorii de bază. Meta-etichetarea unei instanțe indică încrederea în clasificarea făcută de acesta, dacă instanța este clasificată corect de către acel clasificator dintre toți ceilalți clasificatori utilizați. Regula de clasificare a meta-clasificatorului este ca fiecare clasificator de bază să atribuie o clasă la instanța curentă și apoi meta-clasificatorul să decidă dacă clasificarea este demnă de încredere sau nu. Pe lângă creșterea acurateții de clasificare, prin exploatarea sinergismului mai multor clasificatoare, un alt avantaj al meta-clasificării constă în posibilitatea de exploatare a paralelismelor funcționale (multiprocesor).

## 2 Obiectivele proiectului pentru anul 2011

Etapa	Obiective	Activități	Categoriile de buget (Valoare lei)
Unică	1. Design Space Exploration pentru meta-clasificatorul neadaptiv	1.1 Realizarea unei explorări a spațiului problemei folosind algoritmi de căutare euristica.	16500,00
		1.2 Implementarea algoritmilor euristici și găsirea celor mai bune valori de ponderare. Realizare practica	14540,00
	2. Implementarea unui meta-clasificator adaptiv bazat pe rețele neurale	Reprezentarea vectorului documentului de intrare prin agregarea vectorilor de clase generați de către fiecare clasificator în parte.	15000,00
		Dezvoltarea unui meta-clasificator neuronal adaptiv. Realizare practica.	17000,00
<b>Total etapa: 63040,00</b>			

## 3 Sinteza activităților de cercetare realizate în 2011

### 3.1 Setul de date utilizat în experimente

Experimentele prezentate sunt efectuate folosind colecția de date Reuters-2000 [Reut00], pe un set de 7053 documente, care au fost împărțite în mod aleatoriu în setul de antrenament (4702 eşantioane), numit în continuare setul A1, și setul de test (2351 eşantioane), numit în continuare setul T1, care a fost prezentat mai în detaliu în sinteza pe anul 2010.

### 3.2 Meta-clasificator - Arhitecturi neadaptive propuse și dezvoltate

Meta-clasificatorul propus în continuare conține cei 9 clasificatori prezentați în sinteza din anul 2010 și pleacă de la premisa că nu contează doar clasa învingătoare ci toate clasele întoarse de

către meta-clasificator și locul pe care apare fiecare clasă în parte. De exemplu, în cazul a doi clasificatori și 3 clase, dacă o clasă apare o dată pe locul 1 și o dată pe locul 4 și o altă clasă apare de 2 ori pe locul 2, este posibil ca cea de-a doua clasă să fie mai valoroasă, chiar dacă nu a obținut niciodată locul 1.

### 3.2.1 Meta-clasificator neadaptiv bazat pe sumă

În meta-clasificatorul dezvoltat sunt incluși 8 clasificatori de tip SVM (Support Vector Machine) și unul de tip Naive Bayes. Fiecare dintre aceștia produce la ieșire un vector care conține 16 scalari, vezi Fig. 3.1. Fiecare scalar reprezintă valoarea funcției de decizie a clasificatorului pentru clasa respectivă. Până acum, în [Mora08] se alegea întotdeauna valoarea cea mai mare și clasa corespunzătoare a acestei valori era considerată clasa pe care o prezice clasificatorul respectiv. Pentru fiecare document în parte vom obține acum 9 astfel de vectori fiecare cu 16 valori.

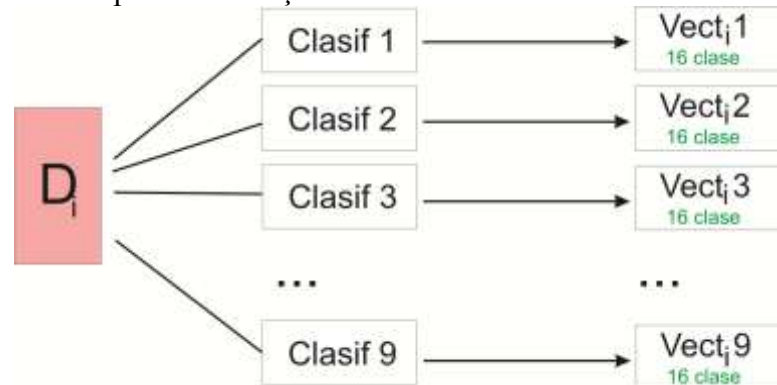


Fig. 3.1 Meta-clasificator neadaptiv bazat pe sumă

Valorile funcțiilor de decizie pentru clasificatorii de tip SVM se află în intervalul  $(-\infty, \infty)$  dar în apropierea valorii 0, iar pentru clasificatorul de tip Bayes valorile se află în intervalul  $(-\infty, 0)$ . Având în vedere aceste diferențe și pentru a putea realiza însumarea valorilor vectorilor, am transpus valorile vectorilor în intervalul  $[1, \infty)$ .

$$V_i = V_i + |\min(\vec{V})| + 1 \quad (3.1)$$

Astfel, pentru fiecare vector, diferențele între valorile vectorilor de ieșire ai clasificatorilor de tip SVM se păstrează. La fel și pentru clasificatorul de tip Bayes. Pentru a putea realiza însumarea acestor vectori, în următorul pas am normalizat vectorii, aducând valorile acestora în intervalul  $(0,1]$ .

În cazul meta-clasificatorului care realizează doar sumele (numit în continuare M-SUM), am însumat cele 16 valori ale acestor 9 vectori, vezi Fig. 3.2, clasa câștigătoare fiind clasa cu valoarea cea mai mare obținută.

$$Class = \max_{c_i, i=1,16} \sum_{k=1}^9 V_i[k] \quad (3.2)$$

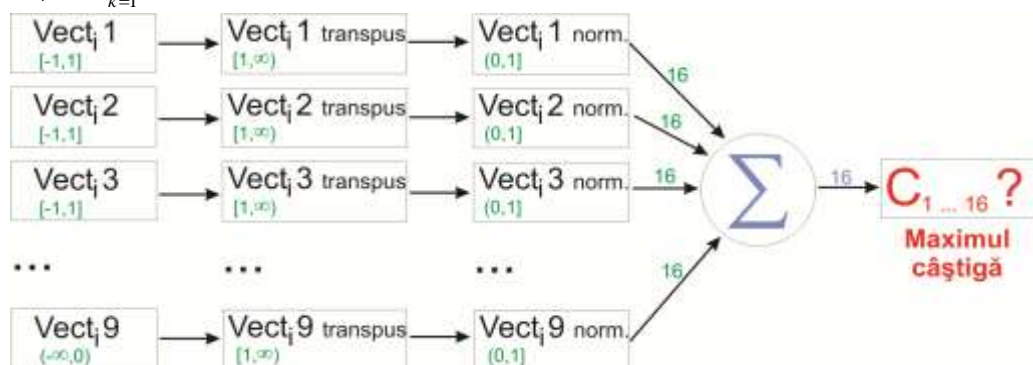


Fig. 3.2 Meta-clasificator neadaptiv bazat pe sumă (M-SUM)

Acest meta-clasificator bazat pe sumă, fiind unul neadaptiv, va obține întotdeauna același rezultat pentru o anumită instanță de intrare. În cazul rulării pe cele 2351 documente de test (setul T1), am obținut un număr de **313** documente clasificate eronat, care reprezintă o acuratețe a

clasificării de **86.68%**, cu **0.59%** mai mare decât valoarea obținută, folosind votul majoritar și toți cei 9 clasificatori. Astfel putem concluziona că metoda bazată pe luarea în considerare doar a clasei învingătoare (*majority vote*), are dezavantaje față de metoda prezentată mai sus. În acest caz există șansa ca o clasă care poate niciodată nu a obținut locul 1, dar a obținut valori apropiate de maxim, să fie în final clasa corectă.

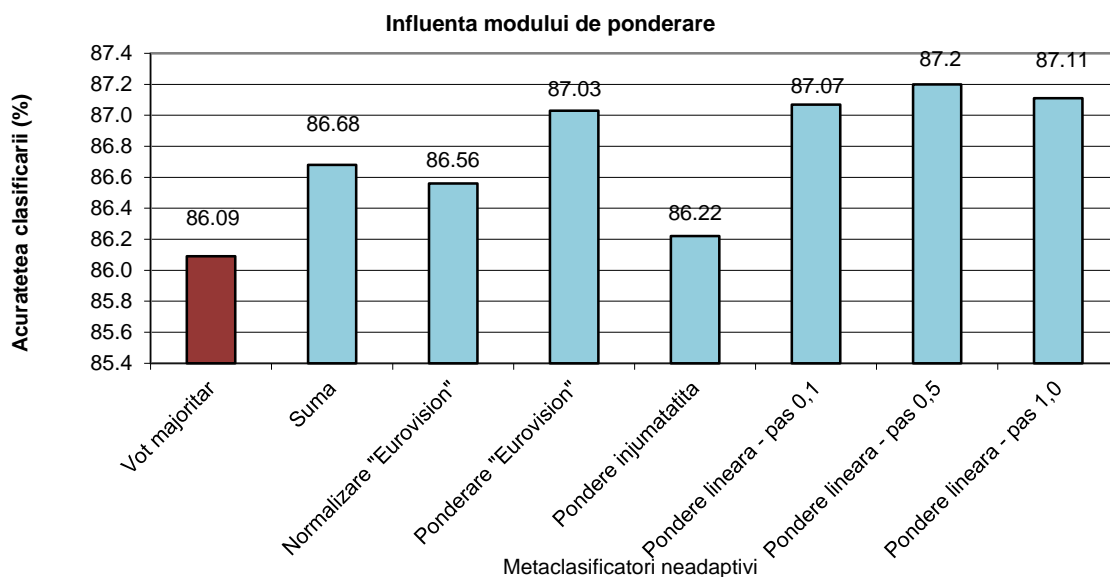
Am realizat o serie de cercetări privind diferite ponderări a elementelor vectorilor. În cazul primei ponderări testate vom asigna în fiecare vector pentru clasa de pe locul 1, valoarea 12, pentru clasa de pe locul 2, valoarea 10, iar în continuare, pentru fiecare clasă de pe următoarele locuri, valori descrescătoare până la valoarea 1 (abordarea implementează soluția consacrată în concursul european de muzică ușoară EUROVISION). În urma acestei ponderări am obținut un număr de 316 erori de clasificare, ceea ce reprezintă o acuratețe a clasificării pe setul T1 de **86,55%** pentru această metodă. Rezultatele obținute sunt cu 0,12% mai slabe decât cele obținute direct pe sumă.

O altă schemă testată este cea în care în pasul în care se realizează ponderarea, am decis că în fiecare vector reprezentat ca în secțiunea anterioară, pentru clasa de pe primul loc, noua valoare să fie vechea valoare **înmulțită** cu 12. Pentru clasa de pe locul 2 va fi vechea valoare înmulțită cu 10...

În acest caz am obținut un număr de **305** documente clasificate eronat pe setul T1, acuratețea clasificării pentru acest meta-clasificator fiind de **87.02%**. Această acuratețe de clasificare obținută este cea mai mare obținută prin utilizarea unui meta-clasificator neadaptiv, dar evident mai mică decât limita maximă de 98.63%, la care poate ajunge teoretic meta-clasificatorul.

Am realizat o serie de cercetări privind și alte variante de ponderare a elementelor vectorilor, cum ar fi înjumătățirea ponderilor, ponderi mici liniar descrescătoare iar rezultatele comparative sunt prezentate sintetic în Fig. 3.3. Totuși, cele mai bune rezultate le-am obținut în cazul în care valorile ponderilor scad liniar cu un pas egal cu valoarea 0,5. Valoarea de prima poziție va fi ponderată cu valoarea 8.5 ș.a.m.d., descrescător până la ultima poziție, unde valoare ponderii este 1.0. Astfel, numărul de documente incorect clasificate de către meta-clasificator a scăzut la 301, rezultând o acuratețe a clasificării de **87.20%**.

Rezultatele prezentate în această secțiune au fost de asemenea publicate în [Cret09].



**Fig. 3.3 - Comparație rezultate metaclasificatori neadaptivi**

### **3.2.2 Meta-clasificator cu ponderi calculate. Design Space Exploration utilizând algoritmi genetici. Rezultate obținute.**

Întrucât alegerea valorilor optime pentru ponderarea fiecărei poziții de clasă este o sarcină dificilă, în această secțiune prezint o metodă bazată pe un algoritm genetic pentru calculul ponderilor optime. Pornind de la setul de date de test, am creat un fișier de testare adecvat, care este folosit pentru algoritmul genetic. Astfel, pentru fiecare document din setul de testare am salvat toate

ieșirile din toți clasificatorii separat. Astfel, pentru un document, am obținut 9 vectori de ieșire (pentru că avem 9 clasificatori), fiecare vector cu un număr de 16 scalari (pentru că avem 16 clase). Elementele fiecăruia dintre cei 9 vectori sunt în prealabil ordonate descrescător. În algoritmul genetic folosit în experimente, un cromozom reprezintă valoarea ponderilor pentru fiecare clasă separat, în funcție de poziția clasei. Prima valoare din vector reprezintă ponderea clasei de pe prima poziție din clasificator, a doua valoare din vector reprezintă ponderea clasei de pe poziția următoare, și așa mai departe. Astfel cromozomul va avea forma:

$$c = (w_1, w_2, \dots, w_j), \text{ cu } j = \overline{1,16} \quad (3.3)$$

unde  $w_j$  reprezintă ponderea pentru fiecare poziție a clasei, returnată de fiecare clasificator utilizat în meta-clasificator. Setul de antrenament este de forma  $\{ \langle \vec{x}_{ij}, y \rangle, i = \overline{1,m} \text{ and } j = \overline{1,n} \}$ , unde  $y$  reprezintă clasa corectă pentru documentul  $\vec{x}_{ij}$ ,  $n$  reprezintă numărul claselor iar  $m$  reprezintă numărul clasificatoarelor utilizate în meta-clasificator. Pentru algoritmul genetic, setul de antrenament este alcătuit din ieșirile tuturor clasificatorilor utilizați în meta-clasificator. Pentru a simplifica calcularea valorii funcției de fitness, voi folosi o reprezentare a cromozomului care utilizează toate ieșirile clasificatorilor utilizați în meta-clasificator. Prin această abordare se va ține cont și de poziția fiecărei clase în ieșirea fiecărui clasificator. În cromozom sunt salvate, pentru fiecare poziție a clasei, ponderile care se vor utiliza în calcularea acurateții meta-clasificatorului. Astfel, decizia meta-clasificatorului include și ponderile pentru fiecare clasă.

Pentru fiecare experiment am pornit cu o generație de 100 de cromozomi, fiecare dintre ei având inițial valori aleatoare cuprinse în intervalul  $[-1, 1]$ . Pentru a genera o populație nouă, am folosit operatorii genetici ca selecția, crossover și mutația. Procesul se oprește după un număr predefinit de 1000 de generații, sau dacă nu apar modificări ale celui mai bun cromozom în ultimele 20 de generații. Pentru fiecare cromozom, funcția de fitness se va calcula ca fiind acuratețea clasificării obținută de meta-clasificator pe setul de test.

Deoarece ordinea claselor este importantă, în cromozom am respectat condiția  $w_1 > w_2 > \dots > w_{16}$  pentru formula (3.3) și această condiție o considerăm cunoștințe de domeniu. Pentru calcularea funcției de fitness, am efectuat următorii pași:

1. pentru fiecare document din setul de test, algoritmul obține un vector de ieșire corespunzător pentru fiecare clasificator inclus în meta-clasificator;
2. fiecare vector de ieșire este ponderat cu valorile corespunzătoare din cromozom;
3. clasa învingătoare se stabilește însumând rezultatele pentru fiecare clasă în parte și alegând-o pe cea cu valoare maximă conform formulei :

$$Class_j = \sum_{i=1}^9 w_j c_{ij}, \text{ for } j = \overline{1,16} \text{ si } WinClass = \arg \max_{j=\overline{1,16}} (Class_j) \quad (3.4)$$

unde  $w_j$  reprezintă valoarea cromozomului pentru ponderea clasei situată pe poziția  $j$ ,  $c_{ij}$  reprezintă valoarea clasei calculată de clasificatorul  $i$  de pe poziția  $j$ ,  $Class_j$  reprezintă valoarea calculată pentru clasa de pe poziția  $j$ .

4. clasa declarată învingătoare (WinClass) de către meta-clasificator va fi comparată cu clasa reală propusă de Reuters iar dacă clasele pentru același document sunt identice, atunci am considerat că documentul a fost clasificat corect;
5. după procesarea tuturor documentelor, valoarea funcției de fitness se calculează ca fiind acuratețea obținută pe setul de testare.

Am considerat ca fiind cel mai bun cromozom acela cu care s-a obținut cea mai mare valoare a funcției de fitness. Ca și metode de selecție a cromozomilor din cadrul unei populații, am folosit două metode: metoda „Roulette Wheel” și metoda lui Gauss. Pentru crearea unei populații am folosit toți cei 3 operatori genetici astfel: în 30% din cazuri am folosit operatorul de selecție (selectând întotdeauna cel mai bun cromozom), în 40% din cazuri am folosit operatorul de mutație iar operatorul de crossover l-am folosit în 30% din cazuri. Totodată am respectat condiția inițială pentru valoarea ponderilor ca  $w_1 > w_2 > \dots > w_{16}$ . Pentru operatorul de crossover am căutat punctul de secționare ( $s$ ), astfel încât condiția  $w_s^{\text{primul\_părinte}} > w_{s+1}^{\text{al\_doilea\_părinte}}$  este satisfăcută. Pentru operatorul de

mutație, după alegerea aleatoare a punctului de mutație ( $m$ ), noua valoare este aleasă aleator din intervalul  $w \in (w_{m-1}, w_{m+1})$ . Aceste condiții au fost impuse în algoritmul genetic, deoarece considerăm că un cromozom care nu le respectă nu reprezintă un cromozom valid și astfel el nu ar trebui să fie generat.

Testele au fost efectuate pe seturile de date utilizate și de către meta-clasificatorii propuși. Pentru fiecare metodă de selecție a cromozomilor, am efectuat câte 4 teste, având în vedere faptul că rezultatul meta-clasificatorului depinde de valorile cu care se inițializează ponderile pentru cele 16 clase. Rezultatele obținute pentru metoda „Roulette Wheel” le prezint în Fig.3.4.

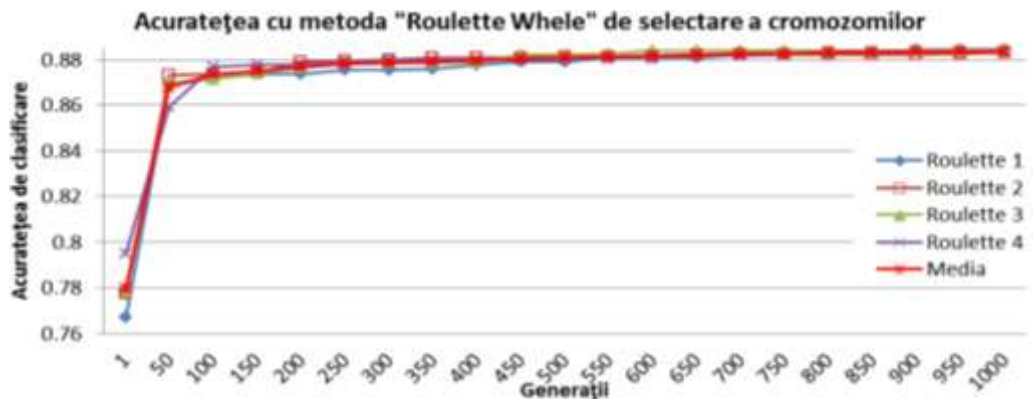


Fig. 3.4 Acuratețea obținută de metaclasificator utilizând metoda ruletei de selectare a cromozomilor

Rezultatele obținute de meta-clasificatorul cu ponderi calculate s-au îmbunătățit în medie față de rezultatele meta-clasificatorului neadaptiv prezentat anterior cu 1.16%; în cazul utilizării metodei „Roulette Wheel” de selecție a cromozomilor, acuratețea de clasificare ajunge la 88.36% iar în cazul utilizării metodei de selecție Gauss, acuratețea de clasificare crește cu 1.17% și ajungând la 88.37%. Cel mai bun rezultat al acestui meta-clasificator a fost obținut în cazul metodei Gauss de selecție a cromozomilor, acuratețea de clasificare ajungând la **88.55%** în acest caz. Rezultatele prezentate în această secțiune sunt prezentate detaliat în [Cret11].

### 3.3 Meta-clasificator adaptiv bazat pe rețea neuronală

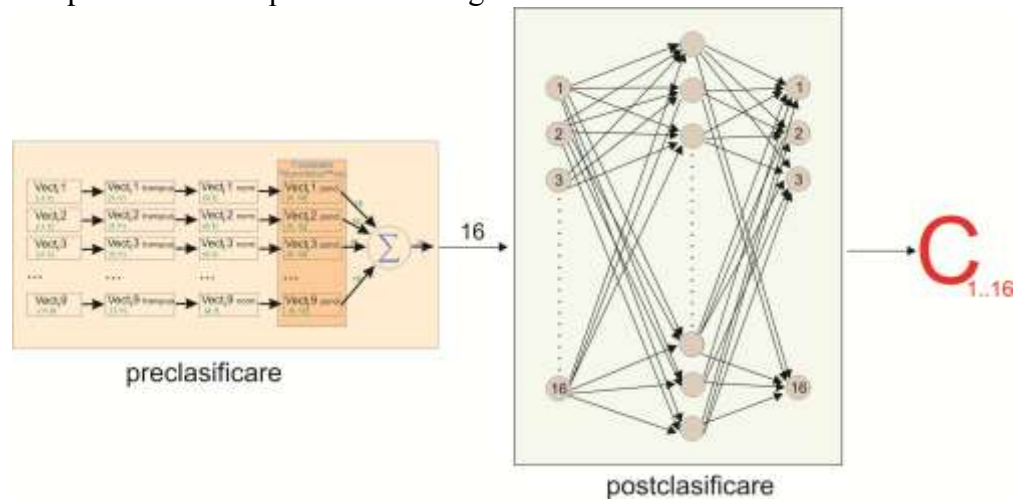
Meta-clasificatorul dezvoltat în continuare se bazează pe o preclasificare și o rețea neuronală de tip feed-forward cu învățare online. Am dorit să includ în meta-clasificatorul numit în continuare **MC-BP (Meta-clasificator cu rețea Backpropagation)** o rețea neuronală, deoarece am considerat că un meta-clasificator adaptiv poate reuși să se „adapteze” și la datele cu probleme, care există în setul de antrenare/testare. Rețelele neuronale sunt sisteme care se adaptează la schimbările survenite în seturile de date, astfel că meta-clasificatorul MC-BP devine unul mult mai adaptiv decât metodele SBDE și SBCOS dezvoltate și prezentate în [Mora06].

Pentru antrenarea și testarea rețelei neuronale cu învățare Backpropagation, am plecat de la setul de vectori obținut de meta-clasificatorul neadaptiv prezentat anterior. Am antrenat acel meta-clasificator cu setul de antrenament A1 (4702 documente) și l-am testat folosind setul de test T1 (2351 documente). Ca și intrare în acest meta-clasificator, avem setul de date iar la ieșire obținem un set de vectori, câte un vector pentru fiecare document de intrare, de 16 elemente fiecare. Setul de vectori obținut, pornind de la setul de documente de antrenare A1, pe care îl vom numi în continuare setul AV1 este un vector agregat conținând 16 elemente (scalari). Acesta va fi folosit în etapa de antrenare a rețelei. Setul de vectori obținut pornind de la setul de documente de testare T1, numit în continuare TV1, va fi folosit atât în etapa de testare cât și în etapa de determinare a configurației rețelei.

În ceea ce privește arhitectura rețelei backpropagation, am ales una care conține două straturi de unități cu funcția sigmoidală de activare, iar fiecare unitate de pe fiecare strat este conectată cu toate unitățile de pe stratul precedent. Deoarece la intrare avem la dispoziție vectori agregați de 16 elemente, rețeaua va avea pe stratul de intrare 16 neuroni. La ieșire, meta-

clasificatorul trebuie să „prezică” clasa în care se găsește documentul curent. Atunci rețeaua Backpropagation va avea la ieșire tot un număr de 16 neuroni, deoarece avem 16 clase distincte, iar la un moment dat va fi activ doar un neuron. În stratul ascuns avem un număr variabil de neuroni, alegerea acestui număr va fi făcut în funcție de rezultatele simulărilor care vor fi prezentate în secțiunea următoare.

În faza de antrenare, deoarece rețeaua este una cu învățare supervizată, pentru setul de antrenare am creat un set cu răspunsurile corecte pentru fiecare document în parte. Un astfel de răspuns conține valoarea „1” pe poziția clasei corecte și valoarea „0” în rest. Structura metaclasificatorului adaptiv M-BP este prezentată în Fig. 3.5.



**Fig.3.5 Metaclasificator adaptiv M-BP**

Deoarece nu există o formulă matematică pentru calcularea numărului optim de neuroni necesari pe stratul ascuns, am realizat câteva experimente în vederea determinării numărului optim de neuroni pentru rețeaua prezentată în Fig. 3.5. Ca și metodă de evaluare, am oprit antrenarea rețelei după ce aceasta a ajuns din punct de vedere al erorii de antrenare la anumită valoare, am evaluat rețeaua pe întreg setul, din punct de vedere al numărului de documente incorect clasificate, după care am continuat antrenarea. Valorile erorii la care am oprit antrenarea rețelei sunt calculate ca fiind sumă a tuturor erorilor obținute pentru fiecare exemplu în parte din setul TV1. Evaluarea rețelei în acel punct se face prin contorizarea numărului de documente incorect clasificate de către rețea. Având în vedere că setul TV1 conține 2351 vectori și că eroarea pe fiecare vector reprezintă o sumă de 16 elemente, eroarea totală va avea valori supraunitare. Vom începe testarea pornind de la o valoare a erorii totale egală cu 500, ceea ce înseamnă o eroare medie pe fiecare vector de 0.21. Ideea este de a ajunge cu eroarea de antrenare la o valoare cât mai mică, într-un timp cât mai scurt.

În cazurile în care rețeaua este mai simplă (are un număr mai mic de neuroni pe stratul ascuns), de la un moment dat eroarea totală a început să scadă foarte încet, moment în care am oprit antrenarea rețelei. De aceea, din acel punct nu vor mai fi valori în graficele pe care le prezint. Am modificat numărul de neuroni utilizând multipli ai lui 16 (Fig. 3.6). Pe măsură ce am crescut numărul neuronilor de pe stratul ascuns păstrând pas de învățare  $\eta=1$ , eroarea a scăzut de la 0.2 la 0.04 per exemplu. Cea mai bună valoare a acurateții de clasificare obținută până în acest moment este de 94.26%, fiind deja superioară celei mai bune valori obținute cu meta-clasificatorul de tip SBCOS cu 9 clasificatoare (93.32%) [Cret10].

Rezultate obținute în cazul antrenării pe setul AV1 și a testării pe TV1 sunt prezentate în Fig. 3.6. Prezint aici rezultate doar pentru arhitecturi ale rețelei cu un număr de neuroni mai mare de 96 pe stratul ascuns și un coeficient de învățare descrescător în timp. Și în acest caz, pentru testare, oprim rețeaua în momentul în care atinge un anumit prag al erorii de antrenare, o testăm pentru a obține numărul de documente incorect clasificate, după care continuăm cu antrenarea. În acest caz, eroarea totală de antrenare este obținută ca o sumă a tuturor celor 4702 erori, ceea ce reprezintă o medie a erorii per exemplu de 0.11 în cazul erorii totale egale cu 500. În acest experiment am ajuns la o eroare totală egală cu 80, ceea ce înseamnă o eroare medie de 0.017 per exemplu.

Arhitectura cu 176 de neuroni pe stratul ascuns a obținut cele mai multe valori minime pentru numărul de documente incorect clasificate, dar în momentul în care eroarea totală de antrenare a scăzut sub valoarea 100, rezultatele cele mai bune au fost obținute de arhitectura cu 192 de neuroni pe stratul ascuns. În acest caz am obținut un număr de 14 documente incorect clasificate, ceea ce reprezintă o acuratețe de clasificare a meta-clasificatorului de **99.40%**. Diferența față de cea mai bună valoare față de cea cu 176 de neuroni pe stratul ascuns este de doar **3** documente incorect clasificate.

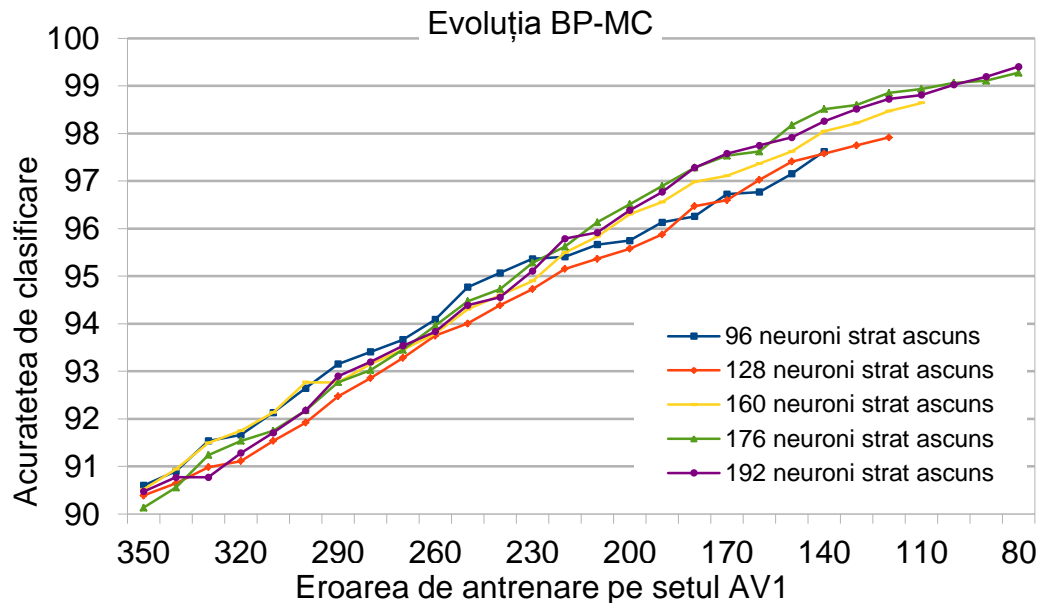


Fig. 3.6 Acuratețea de clasificare în cazul antrenării pe setul AV1 și a testării pe setul TV1

Acest nou meta-clasificator cu o rețea neuronală cu număr suficient de mare de neuroni pe stratul ascuns a reușit să depășească și limita maximă de 98.63% la care ar fi putut ajunge „teoretic” clasificatorii incluși în cadrul meta-clasificatorului.

Foarte interesant, acest meta-clasificator neuronal cu învățare supervizată a demonstrat faptul că acuratețea de 98.63% nu este de fapt limita maximă a meta-clasificării, așa cum eu considerasem. Datorită procesului de învățare supervizată, această limită poate fi depășită. Spre exemplu, în cazul unui vector de intrare în rețea al cărui element maxim nu se află situat pe poziția clasei corecte, acesta poate activa la ieșire celula corectă tocmai datorită unui proces de învățare adecvat (în care rețelei i s-au mai livrat exemple asemănătoare). Această limită maximă a acurateții clasificării este corectă doar pentru metode de agregare neadaptive (postclasificare trivială) ale clasei optimale. Cu acest adaus, limita teoretică rămâne corectă și devine acum clar de ce algoritmul neuronal (adaptiv) o poate depăși și o chiar depășește!

Rezultatul obținut de acest meta-clasificator pentru acuratețea clasificării de **99.40%** este cel mai bun rezultat obținut în toate experimentele efectuate în prezenta lucrare (pentru cazul antrenării și testării pe seturi diferite).

Rezultatele prezentate în aceasta secțiune au fost prezentate și în [Cret10].

## 4 Concluzii

Pornind de la meta-clasificatorul cu 9 clasificatori prezentat în sinteza din 2010 am creat o serie de meta-clasificatori neadaptivi care folosesc diferite procedee pentru ponderarea valorilor generate de către fiecare clasificator în parte, cu scopul de a îmbunătăți acuratețea finală a clasificării. Astfel s-a creat un meta-clasificator care, în loc să contorizeze clasa prezisă de fiecare clasificator în parte, cum ar fi în cazul „vot majoritar” (MV), va însuma simplu toate valorile generate de către clasificatoare pentru fiecare clasă în parte. Am prezentat o serie de experimente care încearcă diferite valori pentru a pondera valorile fiecărei clase din vectorii generați de către clasificatori. Aceste valori ponderează vectorii, în funcție de ordinea obținută de fiecare clasă în cadrul vectorului. Cel mai bun rezultat obținut a fost de **301** documente incorect clasificate, ceea ce



reprezintă o acuratețe a clasificării de **87,20%**. Acest rezultat s-a obținut când s-a utilizat ponderarea liniară cu pasul de „0,5”.

O altă modificare a fost utilizarea de algoritmi genetici pentru găsirea acelor ponderi care, utilizate în meta-clasificatorul neadaptiv prezentat mai sus, să ducă la o îmbunătățire substanțială a acurateței de clasificare. Astfel, am creat un meta-clasificator original cu ponderi calculate, folosind algoritmi genetici. Rezultatele obținute de meta-clasificatorul cu ponderi calculate s-au îmbunătățit în medie, cu **1.16%** în cazul utilizării metodei „Roulette Wheel” de selecție a cromozomilor, acuratețea de clasificare ajungând în medie, la **88,36%**. În cazul utilizării metodei de selecție Gauss pentru selecția cromozomilor dintr-o populație, îmbunătățirea a fost de **1.17%**, ajungând în medie la **88.37%**. Cel mai bun rezultat al acestui meta-clasificator a fost obținut într-un experiment folosind metoda Gauss de selecție a cromozomilor, acuratețea de clasificare ajungând la **88.55%**.

În cazul meta-clasificatorului adaptiv ideea de la care am pornit a fost de a construi acest meta-clasificator format din două componente. O componentă, considerată ca fiind etapa de preclasificare, realizată din meta-clasificatorul (selector) neadaptiv și o componentă nouă, considerată ca fiind etapa de postclasificare, adaptivă, realizată dintr-o rețea neuronală de tip backpropagation. Cele mai bune rezultate (**99.40%** acuratețe de clasificare) s-au obținut folosind o rețea neuronală cu **192** de neuroni pe stratul ascuns. Totuși, din punct de vedere al numărului de rezultate bune obținute pe parcursul antrenării, optimul a fost atins pentru cazul utilizării unei rețele cu **176** de neuroni pe stratul ascuns (chiar dacă în final doar s-a apropiat de eroarea de antrenare minimă). În urma experimentelor efectuate s-a putut observa că introducerea unei rețele neuronale în cadrul meta-clasificatorului face ca acesta să se adapteze mult mai bine la documentele care trebuie clasificate, reușind astfel să clasifice și documentele cu problemă pe care meta-clasificatorii adaptivi și neadaptivi prezentați anterior nu au reușit să le „învețe”. Acest nou meta-clasificator a reușit să depășească și limita maximă de **98.63%** la care ar fi putut ajunge „teoretic” (din cauza algoritmului neadaptiv de agregare finală a clasei optime) clasificarilor incluși în cadrul meta-clasificatorului.

## 5 Referințe bibliografice

- [Cret09] Crețulescu R., Morariu, D., Vintan, L., *Eurovision-like weighted Non-Adaptive Metaclassifier for Text Documents*, The 8<sup>th</sup> RoEduNet International Conference, Galați, Romania, 2009, **indexată (ISI) Thomson Reuters**
- [Cret10] Crețulescu, R., Morariu, D, Vințan, L, Coman, I. – *An Adaptive Metaclassifier for Text document*, 16<sup>th</sup> International Conference on Information Systems Analysis, pp. 372-377, ISBN-13: 978-1-934272-86-2(Collection), ISBN-13: 978-1-934272-88-6(Volume II) ,Florida, USA, 2010 **indexată (ISI) Thomson Reuters**
- [Cret11] Crețulescu, R., Morariu, D., Breazu, M., Vintan L. – *Using Genetic Algorithms for Weights Space Exploration in an Eurovision-like weighted Metaclassifier*, The second international conference in Romania of Information Science and Information Literacy, ISSN 2067-9882, April 2011.
- [Mora06] Morariu, D., Vintan, L., Tresp, V., Feature Selection Method for an Improved SVM Classifier, Proceedings of the 3rd International Conference of Intelligent Systems (ICIS'06), ISSN 1503-5313, vol. 14, pp. 83-89, Prague, August, 2006.
- [Mora08] Morariu, D., *Text Mining Methods based on Support Vector Machine*, Ed. MatrixRom, București, 2008.
- [Reut00] Misha Wolf and Charles Wicksteed - Reuters Corpus: <http://www.reuters.com/researchandstandards/corpus/> lansat în noiembrie 2000, accesat în septembrie 2009