Inteligență artificială – Soft Computing

Preparator Claudia Mihaela Mark
Universitatea "Tibiscus" din Timişoara

ABSTRACT In this paper is presented the artificial inteligence genetic computation concept. "The power of the brain stems not from any single, fixed, universal principle. Instead it comes from the evolution (in both the individual sense and the Darwinian sense) of a variety of ways to develop new mechanisms and to adapt older ones to perform new functions."

1 Soft Computing

Utilizarea pe scară largă a computerelor digitale în procesarea informației a dus la concepția greșită că atât informația cât şi procesarea ei sunt dependente de acestea. Deoarece procesarea informației îşi are originea în modul de prelucrare a datelor de către creierul uman putem considera şi posibilitatea procesării informației de către dispozitive diferite de computerele convenționale.

Cercetarile actuale au ca obiectiv principal realizarea unui astfel de dispozitiv, care să copieze structurile şi principiile de operare ale minții umane.

Computerele digitale s-au dezvoltat rapid dupa 1940 şi deşi au fost proiectate pentru a efectua calcule matematice complexe, şi-au gasit locul şi în domenii precum inteligența artificială.

Totuşi structura fundamentală a computerelor digitale se bazează pe principiul prelucrarii

Secvențiale ceea ce nu se poate spune despre structura sistemului nervos uman, acesta fiind constituit dintr-un număr foarte mare de neuroni, care operează în paralel pentru procesarea diferitelor informații. Deci copiind sistemul nervos uman ar trebui să putem construi o maşină pentru

procesarea paralelă a informațiilor, care să satisfacă nevoia realizării unor calcule complexe într-un timp cât mai scurt.

Ca rezultat al cererii tot mai mari de sisteme complexe şi rapide, nevoia de maşini inteligente a crescut şi totodată au crescut şi eforturile depuse în cercetarea inteligenței artificiale.

Termenul de "soft computing" a fost introdus de Lofti Zadeh, profesor la universitatea Berkley din California, SUA, şi adună sub aceeaşi denumire mai multe ramuri din domeniul computing-ului, cele mai importante fiind rețele neuronale, logica fuzzy şi algoritmi genetici, cu scopul declarat de a "exploata toleranța la imprecizie, nesiguranță şi adevăr parțial, pentru a obține flexibilitate, robustețe şi costuri scăzute".

Aceasta caracteristică îl face să difere fundamental de computing-ul convențional (hard), caracterizat tocmai de lipsa impreciziei şi a adevărurilor parțiale.

Având la bază modelul gândirii umane şi ca scop apropierea de aceasta, soft computing-ul grupează trei domenii aflate nu într-o relație de tip concurențial ci una de complementaritate, în care fiecare partener contribuie cu avantajele şi tehnicile proprii la soluționarea unor probleme imposibil de rezolvat în alt mod.

Astfel rețelele neuronale au capacitatea de a învăța şi de a se adapta, logica fuzzy oferă posibilitatea aproximării, în timp ce algoritmii genetici realizează o căutare sistematizată a soluției optime.

Situat între sistemele de inteligență artificială şi computing-ul convențional, soft computingul reprezintă problema de rezolvat de o aşa manieră încât starea sistemului poate fi măsurată şi comparată cu starea ce se doreşte a fi obținută. Starea sistemului stă la baza adaptării parametrilor, care încetul cu încetul converg către soluția optimă.

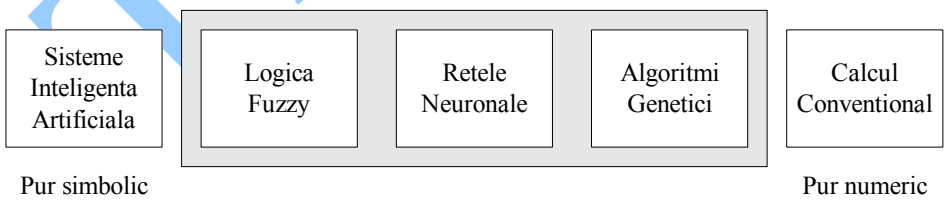

Figura 1.1. Soft computing

Lucrarile lui Zadeh, lucrarea despre seturi fuzzy - 1965, lucrarea despre analiza sistemelor complexe şi a proceselor de decizie – 1973 şi lucrarea despre teoria posibilității şi analiza soft a datelor - 1981, au condus

la naşterea acestui nou domeniu din cadrul domeniului mai vast al inteligenței artificiale. Includerea rețelelor neuronale şi a algoritmilor genetici în soft computing s-a produs mai târziu.

2 Fuzzy Computing

Logica fuzzy a fost aplicată cu succes în multe domenii în care abordarea obişnuită, bazată pe modele, este dificilă sau nu se justifică costurile implementării ei. Oricum, pe măsură ce complexitatea sistemelor create, sunt greu de determinat reguli fuzzy şi funcții de apartenență demne de încredere, folosite la descrierea comportării sistemului. În plus, datorită naturii dinamice a aplicațiilor economice şi financiare, regulile şi funcțiile de apartenență trebuie să se adapteze schimbărilor mediului pentru a continua să fie folositoare.

Principalul avantaj al folosirii rețelelor neuronale în modelarea sistemelor economice sau financiare este că ele pot fi sintetizate fără a folosi cunoştințe detaliate şi explicite ale procesului modelat. Totuşi, date de antrenare limitate sau "zgomotoase" pot cauza ieşiri inconsistente, fără sens. Aceasta este o problema serioasă a rețelelor neuronale.

Din cauza naturii lor complementare, aceste două tehnologii pot fi integrate în mai multe moduri, pentru a se putea trece peste neajunsurile fiecăreia în parte. Un sistem hibrid neurofuzzy e descris în acest articol cu scopul de a explica această integrare.

Sistemele hibride neurofuzzy combină avantajele sistemelor fuzzy, care se ocupă de cunoaşterea explicită, ce poate fi înțeleasă şi explicată, şi cele ale rețelelor neuronale, care se ocupă de cunoaşterea implicită, care poate fi căpătată prin învățare. Învățarea rețelelor neuronale este o bună metodă de ajustare a cunoştințelor expertului şi de generare automată de reguli fuzzy şi funcții de apartenență adiționale, care să se conformeze anumitor specificații; în plus, reduce timpul de proiectare şi costurile. Pe de altă parte, logica fuzzy îmbunătățeşte capacitatea de generalizare a sistemului de rețele neuronale, punând la dispoziție ieşiri mai sigure atunci când este nevoie de extrapolare dincolo de limitele datelor de antrenare.

Sistemul neurofuzzy constă din componentele ce alcătuiesc un sistem fuzzy tradițional, dar fiecare "stage" este efectuat de către un strat de neuroni ascunşi, şi astfel sistemul câştigă capabilitatea de a învăța a rețelelor neuronale.

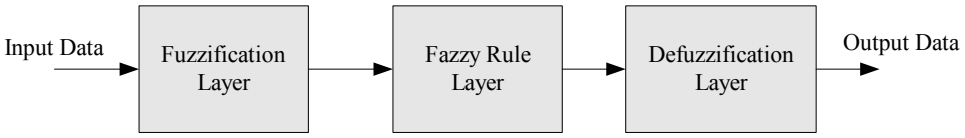

Figura 2.1. Schema arhitecturii unui sistem neuro – fuzzy;

Fiecare neuron din stratul de fuzzificare reprezintă o funcție de apartenență de intrare a unei reguli fuzzy. O metodă obişnuită de a implementa acest strat este să se exprime funcțiile de apartenență ca puncte discrete. Fiecare din nodurile ascunse este definit ca un punct de referință fuzzy în spațiul de intrare. Această metodă poate aproxima multe funcții continue şi gradul de eroare depinde foarte mult de numărul de puncte discrete folosite.

O altă abordare, mult mai bună, este aceea de a folosi o combinație de una sau două funcții sigmoidale şi o funcție liniară pentru a reprezenta fiecare funcție de apartenență în straturile de fuzzificare şi defuzzificare. Parametrii acestor neuroni pot fi antrenați pentru a regla fin forma şi locația finală a funcției de apartenență. În cele mai multe scheme, numărul de neuroni din acest strat e fixat, dar există posibilitatea de a adăuga sau şterge aceşti neuroni în timpul antrenării, în funcție de ieşirile produse de intrările de antrenare. În figura 2.2, fiecare mic dreptunghi reprezintă un neuron.

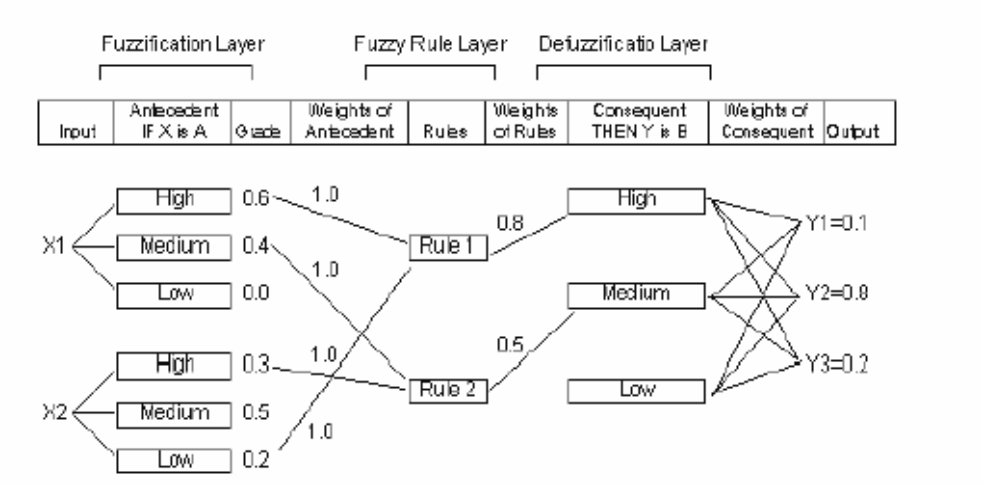

Figura 2.2. Implementarea regulilor Fuzzy

Stratul regulilor fuzzy reprezintă baza de reguli fuzzy şi funcția sa este să execute operațiile logice fuzzy. Fiecare neuron reprezintă o regulă fuzzy

şi calculează certitudinea fiecărei propoziţii compuse care indică potrivirea, adică cât de bine ipotezele fiecărei reguli fuzzy sunt satisfăcute.

Neuronii au o funcţie liniară, iar ieşirile lor sunt conectate la stratul de defuzzificare prin legături ponderate. Ponderile acestor legături reprezintă importanţa relativă a regulilor asociate cu neuronii. Valorile lor pot fi iniţializate de expert sau setate la 1.0, şi apoi antrenate pentru a reflecta importanţa reală în cadrul funcţiilor de apartenenţă conţinute în stratul de defuzzificare.

Funcţia stratului de defuzzificare este să evalueze regulile. Fiecare neuron din acest strat reprezintă o consecinţă, iar funcţia lui de apartenenţă poate fi implementată combinând una sau două funcţii sigmoidale şi funcţii liniare. Certitudinea fiecărei consecinţe este calculată, şi este privită ca potrivirea acelor reguli fuzzy care au aceeaşi consecinţă.

Ponderea fiecărei legături de ieşire a acestor neuroni reprezintă centrul de greutate al fiecărei funcţii de apartenenţă de ieşire a consecinţei, şi este antrenabilă. Valoarea finală de ieşire este apoi calculată folosind metoda centrului de greutate.

Structura din figura 2.2 poate fi configurată cu valorile iniţiale specificate de experţi umani, şi apoi reglată prin folosirea unui algoritm de antrenare, cum ar fi Backpropagation, după cum urmează:

- *Pas 1:* Pentru o mostră de date de intrare, se calculează ieşirea corespunzătoare.
- *Pas 2:* Se calculează eroarea dintre ieşire şi obiectivul real.
- *Pas 3:* Se ajustează ponderile şi funcţiile de apartenenţă.
- *Pas 4:*La un anumit număr de cicli, se şterg regulile şi nodurile funcţiilor de apartenenţă inutile, şi se adaugă altele noi.
- *Pas 5:* Daca Eroarea > Toleranţa atunci se merge la Pas 1, altfel stop.

Când nivelul de eroare scade sub toleranţa specificată de utilizator, ponderile finale ale interconexiunilor reflectă schimbările în regulile fuzzy şi funcţiile de apartenenţă iniţiale. Dacă ponderea rezultată pentru o regulă e apropiată de zero, regula poate fi eliminată fără ezitare din baza de reguli, de vreme ce este insignifiantă în comparaţie cu altele. De asemenea, forma şi poziţia funcţiilor de apartenenţă din straturile de fuzzificare şi defuzzificare pot fi reglate fin prin ajustarea parametrilor neuronilor din aceste straturi, în timpul procesului de antrenare.

Sistemele neurofuzzy oferă precizia şi capabilitatea reţelelor neuronale, rămânând totuşi uşor de înţeles ca sistemele fuzzy. Cunoştinţele explicite obţinute de la experţi pot fi încorporate cu uşurinţă într-un astfel de

sistem, iar cunoştinţele implicite pot fi învaţate din mostrele de antrenare pentru a îmbunătăţi acurateţea ieşirii. În plus, regulile modificate şi cele noi pot fi extrase dintr-un sistem neurofuzzy antrenat corect, pentru a explica cum sunt rezultatele derivate.

Există multe alte metode de a combina tehnicile neurale şi fuzzy, pentru a îmbunătăţi viteza de învăţare, a ajusta învăţarea etc. De asemenea, tehnologii mai noi cum ar fi algoritmii genetici pot fi integrate pentru a îmbunătăţi şi mai mult performanţele sistemelor hibride.

3 Neuro – genetic computing

Reţelele neuronale artificiale (ANN) şi algoritmii genetici (GA) au intrat recent în atenţie. Reţelele neuronale, pot fi privite, pe de o parte, ca metode multivariabile de analiză analitică neliniară şi sunt recunoscute ca fiind foarte potrivite la recunoaşterea structurilor din cadrul unui set complex de date perturbate de zgomot şi estimarea relaţiilor neliniare între acestea.

Multe studii au arătat că reţelele neuronale au capabilitatea de a învăţa mecanismele ascunse ale seriilor temporale, sau, în cazul aplicaţiilor comerciale, dinamica pieţii. Totuşi, în general, este dificil de proiectat reţele neuronale performante, deoarece multe din principiile de bază care guvernează procesarea informaţiei în reţea sunt greu de înţeles, iar interacţiunile complexe între componentele reţelei nu permit, în cele mai multe cazuri, aplicarea tehnicilor inginereşti de tip “divide-and-conquer”.

Atunci când sunt impuse combinaţii complexe ale criteriilor de performanţă (ca, de exemplu, viteza de învăţare, compactitatea, abilitatea de generalizare şi rezistenţa la zgomot) şi pe măsură ce aplicaţiile reţelei tind să crească în dimensiuni şi complexitate, abordarea uman – inginerească nu va funcţiona, fiind necesară o soluţie mai eficientă, automatizată.

Algoritmii genetici, pe de altă parte, rămân reminiescenţi reproducerii, în care genele a doi părinţi se combină pentru a le forma pe cele ale fiului lor. Când algoritmii genetici sunt aplicaţi la rezolvarea problemelor, premiza de bază este aceea că se poate creea o populaţie iniţială de indivizi reprezentând posibile soluţii la o problemă care trebuie să fie rezolvată.

Fiecare dintre aceşti indivizi va avea anumite caracteristici care-l fac mai mult sau mai puţin potrivit, ca membru al populatiei. Membrii mai potriviţi vor avea o probabilitate mai mare de a se putea reproduce decât membrii mai puţin potriviţi, pentru a produce descendenţi care să aibă o şansă semnificativă de a reţine caracteristicile dorite ale părinţilor lor.

Această metodă este foarte eficientă la găsirea soluției optimale sau apropiate de optim la o varietate de probleme, deoarece nu impune multe dintre limitările care sunt cerute de metodele tradiționale.

Este o strategie elegantă care generează - testează şi poate identifica, exploata regularitățile din mediu şi converge la soluțiile care erau optimale din punct de vedere global, sau foarte apropiate de optim.

Algoritmii genetici s-au aplicat din ce în ce mai mult la proiectarea rețelelor neuronale, în câteva moduri: optimizarea topologiei, algoritmi de antrenare genetici şi optimizarea parametrilor de control. În optimizarea topologiei, algoritmul genetic este folosit pentru a selecta o topologie (număr de straturi ascunse, număr de noduri ascunse, structura de interconectare) pentru rețeaua neuronală, care, la rândul ei, este antrenată folosind o schemă de antrenare clasică, cel mai adesea "back – propagation". In cazul algoritmilor genetici de antrenare, cunoaşterea rețelei neuronale este formulată în termenii problemei optimizării ponderilor, de obicei folosindu-se inversul erorii pătratice medii ca o măsură de apropiere.

Majoritatea parametrilor de control, ca: rata de învățare, rata momentană, nivelul de toleranță, etc., pot fi, de asemenea, optimizați folosind un algoritm genetic. In plus, algoritmii genetici au fost folosiți în multe alte moduri inovative, de exemplu, pentru crearea de noi indicatori pe baza celor existenți, la selectarea indicatorilor buni, la evoluția sistemelor optimale, sau drept tehnică complementară altor metode, cum ar fi logica fuzzy. Algoritmii genetici sunt inspirați din procesul biologic de evoluție al vieții. Fiecare punct din spațiul soluției este codificat ca un şir binar, denumit "cromozom". De exemplu, punctul de coordonate (1, 8, 12) va fi codificat cu şirul:

Figura 3.1. Codificarea într-un şir binar

Există patru stadii în procesul genetic de căutare: inițializarea, evaluarea şi selecția, "crossover"-ul, mutația. In etapa de inițializare, o populație de structuri genetice care sunt distribuite aleatoriu în spațiul soluției este selectată ca punctul de plecare al căutării. In etapa a doua, fiecare structură este evaluată folosind o funcție de potrivire şi îi este asignată o valoare de potrivire (formă bună). Pe baza valorilor de potrivire

relative, structurile din cadrul populației curente sunt selectate pentru reproducere.

O procedură statistică asigură ca numărul aşteptat al descendenților asociat cu o structură dată, *s*, să fie: μ(s)/μ(P), unde μ(s) este performanța observată a lui *s*, iar μ(P)este performanța medie a tuturor structurilor din cadrul populației curente. Din această cauză, structurile de înaltă performanță vor fi alese cu probabilitate mai mare pentru replicare, în timp ce structurile cu o performanță slabă sunt, în cele din urmă, îndepărtate din populație. In absența altui mecanism, de exemplu un proces de selecție, structurile cele mai performante din populația inițială vor tinde să ocupe o proporție din ce în ce mai însemnată din populație, de-a lungul timpului.

Structurile selectate, sunt apoi recombinate, folosind "crossover" (interschimbarea încrucişată a genelor), pentru a maximiza probabilitatea de reținere a caracteristicilor "positive" din generația anterioară. Acest proces este analog reproducerii biologice, unde descendenții pot fi superiori ambilor părinți, dacă moştenesc "genele bune" de la amândoi:

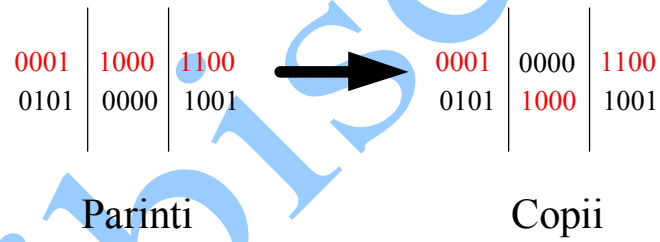

Figura 3.2. "Crossover"-ul genelor între cromozomi

In general, "crossover"-ul se bazează pe prezența informației căutate în cadrul populației curente, la generarea de noi soluții pentru evaluare. Dacă informația specifică lipseşte (datorită limitărilor în stocare sau datorită pierderii în cadrul procesului de selecție al generației anterioare), atunci tehnica "crossover" nu va putea să producă noi structuri care să conțină acea piesă de informație ("genă").

Pentru a rezolva problema, se introduce încă o etapă, numită "mutație". Un operator de mutație, care alterează arbitrar una sau mai multe componente ale unei structuri selectate, furnizează mijlocul de introducere a informațiilor noi în cadrul populației. Totuşi, mutația lucrează ca un operator de fundal, cu o probabilitate foarte mică de aplicare. Prezența

acesteia asigură ca probabilitatea de atingere a oricărui punct din spațiul de căutare să nu fie niciodată nulă.

De exemplu, una din problemele care apar în cazul rețelelor neuronale este că acestea au o anume tendință de a ajunge la soluții suboptimale. Tehnica mutației din algoritmul genetic, furnizează o "scăpare" din această convergență la o soluție suboptimală. Mutația, însă, trebuie menținută la un nivel minim, pentru a evita distrugerea cromozomilor "buni".

O metodă de automatizare a procesului de proiectare a arhitecturii rețelelor neuronale, folosind algoritm genetic este descrisă în continuare. Această metodă presupune utilizarea a două procese adaptive:

- căutarea genetică în fereastra datelor de intrare, în orizontul de predicție, în spațiul arhitecturii rețelei și în spațiul parametrilor de control pentru selecția celor mai buni performeri;
- învățarea prin algoritmul "backpropagation" în rețele individuale, pentru evaluarea arhitecturilor selectate.

Metoda începe cu o populație inițială de rețele generate aleator, care este reprezentată de structurile arborescente suprapuse, ca în figura următoare:

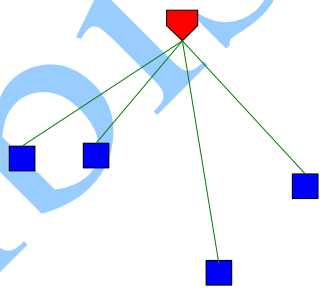

Figura 3.3. Una din rețelele generate aleator

In figura de mai sus, fiecare dreptunghi reprezintă un nod de intrare, iar poligonul reprezintă un nod de ieşire. Rețelele vor creşte, iar nodurile lor ascunse vor fi inserate în interiorul rețelelor, pe măsură ce populația evoluează. Nodurile de intrare preiau o combinație aleatorie a datelor de intrare dintr-o fereastră de intrare care culege articole dintr-un fişier cu date de antrenare şi alimentează cu ele rețeua.

Nodul de ieşire, selectează în mod aleatoriu un orizont de predicție dintr-o fereastră de ieşire şi utilizează articolul de date asociat ca o valoare țintă. Ambele ferestre de date baleiază fişierul cu datele de antrenare în mod

secvențial sau aleatoriu. Figura următoare ilustrează o fereastră de intrare cu dimensiunea 3 şi o fereastră de ieşire cu orizonturile de predicție variind între 1 şi 3 paşi înainte:

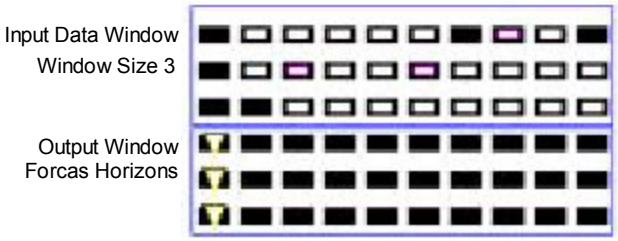

Figura 3.4. Configurațiile inițiale ale ferestrelor de intrare şi ieşire folosite de toate rețelele

Populația inițială de rețele trece apoi prin primul ciclu de evoluție. Ca şi în sistemele biologice reale, ciclurile de învățare sunt imbricate în ciclurile de evoluție în populații. Fiecare ciclu de învățare antrenează întreaga populație de rețele cu setul de perechi intrare-ieşire furnizat de ferestre. Ieşirile rețelelor sunt comparate cu ținta dorită, iar ponderile de interconectare sunt ajustate pentru a obține maparea intrare – ieşire dorită prin minimizarea erorilor. La sfârşitul fiecărui ciclu de învățare, rețelele sunt clasificate potrivit unor criterii predeterminate, cum ar fi capacitatea lor de

generalizare. Rețelele cu performanțe reduse vor fi îndepărtate din cadrul populației, pe când cele mai potrivite sunt reținute şi selectate pentru procesul de "crossover" pentru a produce descendenți pentru următoarea generație. Există câteva variante de a încrucişa rețelele – părinte. O metodă ar fi producerea unui nou descendent prin reproducerea a doi dintre cei mai potriviți părinți, ca în figura următoare:

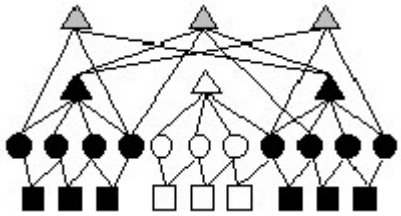

Figura 3.5. Selecția a două dintre rețelele – părinți cele mai potrivite pentru producerea unui nou descendent

In această variantă, nodurile de ieşire ale părinţilor sunt folosite ca noduri ascunse ale descendentului, care va moşteni, astfel, cunoaşterea deja obţinută de către părinţii săi.

Ocazional, se introduce şi mutaţia pentru a evita ca reţelele să nu fie cumva captate într-un minim local în timpul procesului de învăţare. Un mod de a produce mutaţii este împrăştierea aleatorie a ponderilor acelor reţele care au fost clasificate cu performanţe scăzute şi schimbarea combinaţiilor de date în fereastra lor de intrare şi/sau orizontul lor de predicţie.

După fiecare ciclu de evoluţie, se păstrează o imagine a celei mai nimerite reţele. Imaginea include combinaţiile curente de intrare, orizontul de predicţie, structurile ("pattern"-urile) de interconectare şi ponderile, astfel încât cea mai potrivită reţea să poată intra în urmatoarele cicluri de evoluţie şi de învăţare împreună cu restul populaţiei de reţele noi formate. Această imagine va rămâne intactă în timpul următoarelor cicluri succesive de evoluţie până când va apărea o altă reţea mai potrivită.

Teoria selecţiei naturale oferă câteva argumente convingătoare în sensul că indivizi cu anumite caracteristici sunt mai capabili de supravieţuire şi transmit mai departe aceste caracteristici descendenţilor lor. Un algoritm genetic este o procedură generală de căutare bazată pe ideile geneticii şi ale selecţiei naturale, iar puterea sa constă în faptul că, pe măsură ce membri ai populaţiei se împerechează, ei produc descendenţi care au o şansă semnificativă să reţină caracteristicile dorite ale părinţilor, probabil chiar să combine cele mai bune caracteristici ale ambilor părinţi. In această manieră, calitatea totală a populaţiei poate potenţial să crească de la generaţie la generaţie, pe măsură ce se descoperă soluţii mai bune ale problemei.

Când se aplică optimizării reţelelor neuronale pentru problemele de predicţie şi clasificare, algoritmii genetici pot fi folosiţi la căutarea combinaţiilor corecte de date de intrare, a orizontului de predicţie cel mai potrivit, a structurilor şi ponderilor de interconectare optimale sau aproape optimale între neuroni, şi a parametrilor de control (rata de învăţare, rata momentană, nivelul de toleranţă, etc.) optimi, pe baza datelor de antrenare folosite şi a criteriilor predefinite.

Ca şi reţelele neuronale, algoritmii genetici nu garantează întotdeauna o soluţie perfectă, dar, în multe cazuri, pot ajunge la soluţii acceptabile fără timpul şi costul unei căutări exhaustive.

4 Genetic – fuzzy computing

Ingrediente cheie in realizarea unui sistem inteligent sunt structurile de reprezentare a cunoaşterii şi metodele folosite pentru adaptarea acestor structuri la scopul urmărit. Tehnicile hibride aparţinand acestui domeniu exploatează capabilităţile de reprezentare a cunoaşterii ale sistemelor fuzzy şi capabilităţile adaptive ale algoritmilor genetici.

În rândurile ce urmează vor fi prezentate mai multe tehnici a căror esenţă este variaţia multiobiectiv a algoritmilor genetici. Mai întâi se va arăta cum algoritmii genetici multiobiectiv pot fi folosiţi în proiectarea sistemelor fuzzy şi apoi vor fi propuse tehnici de îmbunătăţire a metodelor bazate pe algoritmi genetici prin folosirea sistemelor fuzzy. Tehnicile genetic-fuzzy propuse permit identificări structurale şi parametrice simultane şi optimizări multiobiectiv bazate pe algoritmi genetici.

Optimizarea multiobiectiv are aplicaţii în viaţa reală deoarece oamenii sunt nevoiţi adeseori să aleagă între soluţii cu performanţe diferite, luând în calcul obiective multiple.

De exemplu, în cazul generării cablajului unui circuit integrat, trebuie avute în vedere consideraţii de timing şi de geometrie, şi poate fi necesară realizarea unui compromis. In probleme de această natură există de obicei mai multe soluţii, care nu pot fi diferenţiate din punct de vedere calitativ în absenţa unui criteriu relativ la importanţa elementelor ce trebuie luate în calcul în apreciere. Soluţia aleasă este în ultimă instanţă bazată pe cerinţele unei aplicaţii sau este o soluţie aleasă după un criteriu subiectiv.

În primul pas din cadrul optimizării multiobiectiv se furnizează un set de soluţii care reprezintă cele mai bune alegeri (un set Pareto optimal), iar in al doilea pas se alege o soluţie finală.

Optimalitatea lui Pareto se bazează pe principiul dominanţei. Soluţiile incluse în setul optimal al lui Pareto sunt cele care nu mai pot fi îmbunătăţite într-o anumită direcţie fără a fi deteriorate simultan în celelalte direcţii. In continuare va fi prezentată o tehnică de generare a setului Pareto de soluţii folosind un algoritm evolutionist.

Algoritmii genetici reprezintă strategii stohastice de căutare, modelate după mecanisme caracteristice naturii. Căutarea se bazează pe informaţii ce caracterizează întreaga populaţie. Un individ din cadrul populaţiei codifică o soluţie sub forma unui şir de parametrii, şir asupra căruia se vor aplica operaţiile de mutaţie şi crossover.

Paşii executării unui algoritm genetic sunt următorii:

- Generarea aleatoare a unei populaţii iniţiale;

- Evaluarea potrivirii indivizilor;
- Selectarea părinților pentru împerechere;
- Aplicarea operatorilor de mutație şi crossover şi generarea noii populații;
- Dacă criteriul de stop nu este satisfăcut se reia algoritmul de la pasul 2, în caz contrar execuția algoritmului se încheie.

Operatorul crossover este un mecanism care combină informațiile de la părinți şi le transmite copiilor. De exemplu în genetica umană un copil primeşte un set de gene de la fiecare părinte.

Operatorul mutație schimbă aleator informațiile deținute de o genă, şi are rolul de a reintroduce în sistem anumite caracteristici care altfel s-ar putea pierde definitiv datorită procesului de selecție.

Deseori se folosesc operatori dependenți de aplicație, de exemplu operatorul de crossover bazat pe ordine.

Criteriul de stop este şi el dependent de aplicație şi este satisfăcut când un individ ajunge să încapsuleze anumite caracteristici sau după un anumit număr de generații. Folosirea unui algoritm genetic presupune:

- Găsirea unui reprezentări a soluției;
- Realizarea unei codificări genetice a soluției;
- Găsirea unei funcții de evaluare potrivite.

Performanța unui individ este înglobată într-o singură valoare deşi este foarte greu de inclus toate obiectivele în aceasta.Modelarea fuzzy este un element foarte important in teoria fuzzy.

Metodologia generală a modelarii fuzzy este alcătuită din trei etape:

- Identificarea structurii externe;
- Identificarea structurii interne;
- Aproximarea lingvistică.

Identificarea structurii externe are in vedere determinarea variabilelor de intrare relevante. Identificarea structurii interne implică partiționarea spațiului şi găsirea parametrilor de ieşire.

Aproximarea lingvistică este ultimul pas în modelarea unui sistem fuzzy şi presupune maparea structurii interne în termeni lingvistici (acest pas este adesea omis).

Ne propunem să realizăm combinarea dintre reprezentarea fuzzy a unui sistem şi un algoritm multiobiectiv evoluționist de învățare. Tehnica propusă integrează primele două etape ale modelării fuzzy într-una singură.

O metoda des întâlnită este folosirea sistemelor fuzzy bazate pe reguli. In aceste sisteme spațiul variabilelor de intrare este partiționat în combinații de seturi fuzzy folosind reguli fuzzy de tip if – then.

Proiectarea unui sistem fuzzy poate fi realizată în cinci paşi:

- Determinarea intrărilor relevante;
- Determinarea numărului de reguli;
- Determinarea funcţiilor de apartenenţă;
- Determinarea acţiunilor asociate cu fiecare partiţie;
- Determinarea metodei adecvate folosită pentru combinarea acţiunilor multiple.

Primul pas este de fapt identificarea externă a sistemului, deoarece intrările şi ieşirile relevante sunt externe sistemului. Din descrierea modului în care se formează partiţiile multidimensionale este clar ca funcţiile de apartenenţă şi numărul de reguli sunt strâns legate de partiţionare, din acest motiv paşii 2 şi 3 se vor executa simultan.

5 Arhitectura sistemelor fuzzy "shared – triangular"

Reprezentarea "shared-triangular" foloseşte funcţii de apartenenţă triunghiulare asimetrice şi operatorul de calcul al minimului pentru a sintetiza funcţii multidimensionale de apartenenţă.

Fiecare funcţie de apartenenţă triunghiulară este specificată prin centru, lăţime stânga şi lăţime dreapta. Regulile acoperă spaţiul de intrare prin selectarea şi combinarea funcţiilor de apartenenţă unidimensionale dintr-un set definit global (toate regulile au acces la acelaşi set de funcţii de apartenenţă). Există deasemenea posibilitatea ca o regulă să nu aibă asociată nici o funcţie de apartenenţă ceea ce înseamnă că ea va acoperi în întregime una din dimensiunile de intrare.

Codul genetic este alcătuit din două macrostructuri: gene ale funcţiilor membre şi gene ale regulilor. Genele funcţiilor membre specifică distanţa între centrele triunghiurilor adiacente, şi lăţimile la stânga respectiv la dreapta, în timp ce genele regulilor deţin locaţii pentru indicii seturilor de funcţii de apartenenţă pentru fiecare dimensiune şi un bit de validitate. Numarul total de parametrii într-un sistem cu b dimensiuni, r reguli şi un număr maxim de c funcţii de apartenenţă pentru fiecare dimensiune este egal cu 3*b*c+r*(2*b+3). Fiecare regulă are nevoie de 2*b+3 parametrii deoarece funcţiile de apartenenţă asociate fiecarei dimensiuni trebuiesc specificate, acest lucru implementându-se prin intermediul unei scheme de indexare.

6 Concluzii

Trăim într-o lume pe cale de a deveni complet automatizată şi există o cerere crescândă de maşini care să poată răspunde dinamic la schimbările ce intervin în mediul înconjurător. Soft computing-ul se va dovedi cu siguranță folositor în domeniile unde modul uman de a raționa este absolut necesar, domenii precum construirea de avioane fără pilot, sisteme de recunoaştere, etc.

Bibliografie

[Hea02] **Steve Heath**, *Embedded Systems Design*, Elsevier Science, 2002

[Rus00] **Stuart J. Russell,** *Artificial Intelligence* , Edition: Hardcover 2000

[RY00] **Randall C. O'Reilly, Yuko Munakata,** *Understanding the Mind by Simulating the Brain Embedded Systems Design*, Textbook Paperback, September 2000

[Win92] **Patrick Henry Winston,** *Artificial Intelligence* , March 1992