UNIVERSITATEA "POLITEHNICA" BUCUREŞTI
Facultatea de Electronică şi Telecomunicaţii

Angel Caţaron

REŢELE NEURALE PENTRU RECUNOAŞTEREA VORBIRII

Teză de doctorat

Conducător ştiinţific:
Prof. dr. ing. Victor-Emil Neagoe

BUCUREŞTI
2003

Angel Caţaron

REŢELE NEURALE PENTRU RECUNOAŞTEREA VORBIRII

Teză de doctorat

Conducător ştiinţific:
Prof. dr. ing. Victor-Emil Neagoe

Rezumat

Această lucrare investighează, pe de o parte, *cum poate fi implementat conceptul de concurenţă la nivelul unei colecţii de reţele neurale* şi, pe de altă parte, *modul în care importanţa diferită a intrărilor influenţează performanţele în recunoaştere ale unor tipuri de reţele neurale.* Recunoaşterea formelor, domeniu în care se plasează şi subiectul acestei teze, oferă un câmp foarte larg cercetării. Recunoaşterea vorbirii grupează o gamă diversă de probleme precum recunoaşterea cuvintelor izolate, identificarea vorbitorilor pe baza amprentei lor vocale, recunoaşterea fonemelor etc. Reţelele neurale sunt un instrument care şi-a demonstrat eficienţa în rezolvarea unei palete largi de aplicaţii între care şi cele de recunoaştere automată a vorbirii.

Majoritatea modelelor neurale abordează recunoaşterea formelor ca o problemă unitară, globală, fără a face distincţia între aportul individual diferit al intrărilor.

Este cunoscut că modularitatea şi principiul *divide et impera* aplicate reţelelor neurale le pot îmbunătăţi performanţele. Introducem modelul *Concurrent Neural Networks (CNN)* care îmbină paradigmele învăţării supervizate şi nesupervizate şi oferă o soluţie primei probleme. Ideea concurenţei este folosită la nivelul unei colecţii de reţele neurale care sunt instruite independent pentru a rezolva subprobleme diferite. Recunoaşterea se face prin identificarea reţelei neurale care furnizează cel mai bun răspuns. Aşa cum o demonstrează şi rezultatele experimentale, acurateţea recunoaşterii este mai mare atunci când folosim modelul propus de noi faţă de cazurile în care nu folosim concurenţa.

A doua problemă are conotaţii mai largi deoarece se regăseşte şi în aplicaţii de *data mining.* Prin asocierea unei ponderi fiecărei intrări, se stabileşte intensitatea cu care intrările influenţează răspunsul sistemului şi implicit capacitatea sa de recunoaştere corectă. Dezvoltăm algoritmul *Ordered Weighted Aggregation - Relevance LVQ (OWA-RLVQ)* care calculează ponderile sub forma unui set de coeficienţi *Ordered Weighted Aggregation (OWA).* Algoritmii *Mutual Information Relevance LVQ (MIRLVQ)* şi *Energy Relevance LVQ (ERLVQ)* propuşi în continuare stabilesc aceşti coeficienţi prin maximizarea informaţiei mutuale pe de o parte, şi a unei mărimi definite cu ajutorul energiei informaţionale, pe de altă parte. Avantajele algoritmilor noştri sunt confirmate experimental prin rate de recunoaştere în mod constant superioare faţă de alţi algoritmi care rezolvă acelaşi tip de probleme.

Mulţumesc d-lui Prof. dr. ing. Victor-Emil Neagoe care în calitate de conducător ştiinţific m-a sprijinit constant prin colaborarea fructuoasă de-a lungul acestor ani, prin discuţiile pline de substanţă şi prin sfaturile fără de care nu aş fi putut finaliza această teză.

De asemenea, mulţumesc tuturor colegilor care au influenţat într-un fel sau altul conţinutul acestei teze. Mulţumesc profesorilor care mi-au transmis pasiunea şi dorinţa de a progresa. Multe mulţumiri prietenilor care m-au încurajat să merg mai departe în timpul perioadei de pregătire a tezei.

Mulţumesc părinţilor pentru că prin educaţia pe care mi-au dat-o şi prin modul în care m-au crescut m-au ajutat să ajung până aici.

Mulţumesc familiei care m-a susţinut în toată această perioadă, mi-a oferit suportul moral, motivaţia de care am avut nevoie şi care m-a învăţat ca nimic nu este greu pentru că nu sunt singur.

Braşov, septembrie 2003

Notaţii

Am folosit caractere **aldine** pentru a nota vectorii şi matricile şi caractere *cursive* pentru componentele matricilor şi pentru scalari în general.

Vectorii care apar în text sunt vectori coloană:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

iar

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^t$$

este un vector linie. Prin t am notat matricea transpusă.

Indicii sunt plasaţi în partea dreaptă jos a literei cu care se notează vectorul sau scalarul. În anumite contexte, pentru claritatea expunerii am folosit şi indici superiori însoţiţi de explicaţiile de rigoare pentru a evita confuzia cu operaţia de ridicare la putere. În această relaţie, p este un indice:

$$E^p = \frac{1}{2} \sum_{j \in O} (y_j^p - d_j^p)^2.$$

În expresiile ale căror valori se calculează iterativ, am folosit notaţia (t) pentru a desemna pasul iteraţiei:

$$\Delta w_{ji}^{(t)} = \alpha \Delta w_{ji}^{(t-1)} + \eta \delta_j^{(t)} y_i^{(t)}.$$

Produsul scalar a doi vectori $\mathbf{x}_1$ şi $\mathbf{x}_2$ este:

$$\mathbf{x}_1^t \mathbf{x}_2 = \sum_{i=1}^{n} x_{i1} x_{i2}.$$

Norma euclidiană a unui vector este notată prin:

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^t \mathbf{x}}.$$

Pentru matricea unitate n-dimensională folosim următoarele notaţii:

$$\mathbf{I}_n = \mathbf{I} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}.$$

Preferăm varianta fără indice pentru simplitate, iar dimensiunea matricii se va deduce din context.

Cuprins

Capitolul 1

Introducere

Această lucrare abordează o temă recunoaştere a formelor folosind reţele neurale, cu aplicaţii în recunoaşterea vorbirii. Sunt introduse mai multe modele originale de recunoaştere, iar dezvoltările teoretice şi rezultatele experimentale demonstrează viabilitatea acestora. Spectrul aplicaţiilor de recunoaştere a vorbirii abordate în teză este larg, începând cu recunoaşterea fonemelor, continuând cu recunoaşterea cuvintelor izolate şi terminând cu recunoaşterea vorbitorilor. Bazele de date sunt diverse, fiind vorba atât de colecţii standard pentru limba engleză cât şi de seturi alcătuite special pentru experimentele din această lucrare şi conţin cuvinte din limba română. Am testat unele modele propuse de noi şi pe seturi de date standard în recunoaşterea formelor, cu scopul de a ilustra faptul că ele pot fi folosite şi în alte contexte decât cel al recunoaşterii vorbirii.

Acest capitol descrie principalele domenii care sunt abordate în lucrare, iar în final conţine o prezentare a structurii tezei.

1.1 Recunoaşterea formelor

Înţelegerea cuvintelor vorbite, recunoaşterea unei feţe umane, citirea caracterelor scrise de mână sunt acţiuni fireşti, specific umane, în spatele cărora stau procese complexe care nu sunt încă deslușite, dar pe care dorim să le modelăm prin sisteme de recunoaştere automată. *Recunoaşterea formelor* înseamnă analiza datelor care provin de la o intrare fizică în sistem şi încadrarea acestora în anumite categorii. De la recunoaşterea automată a vorbirii, identificarea amprentelor digitale până la recunoaşterea caracterelor scrise de mână şi identificarea secvenţelor ADN, gama de aplicaţii a recunoaşterii formelor este imensă, iar crearea unor maşini care să realizeze acest lucru automat şi cu acurateţe este extrem de utilă. Pentru multe probleme de recunoaştere eforturile de găsire a

soluţiilor sunt, de fapt, influenţate de cunoştinţele pe care le avem despre modul în care sunt rezolvate acestea de creierul uman. Reţelele neurale artificiale reprezintă un suport care, în ultimii ani, s-a dovedit foarte performant.

Intrarea unui sistem de recunoaştere a formelor (fig. 1.1) este în majoritatea cazurilor un dispozitiv care preia informaţia primară şi o converteşte într-o colecţie de date. Un astfel de dispozitiv poate fi alcătuit dintr-unul sau mai multe microfoane, una sau mai multe camere de luat vederi etc.

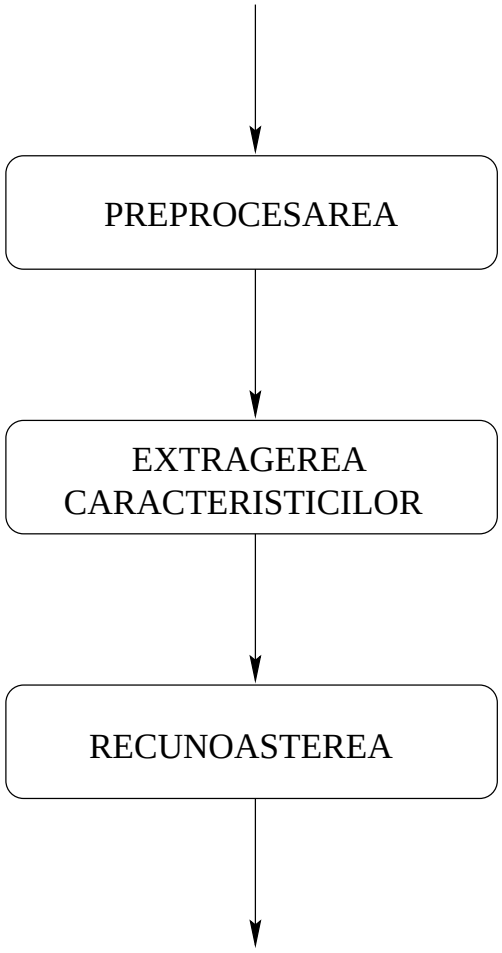

Figura 1.1: Arhitectura unui sistem de recunoaştere a formelor.

Colecţia de date este sursa de informaţie a întregului sistem, iar modulul de preprocesare trebuie să segmenteze datele, adică să stabilească unde începe şi unde se termină fiecare formă. Vom folosi în continuare termenul *formă* care are aceeaşi semnificaţie cu cea a cuvântului *pattern* din limba engleză. Segmentarea este una dintre cele mai dificile probleme în recunoaşterea formelor.

În problemele de recunoaştere automată a vorbirii putem să ne propunem ca segmentarea să însemne găsirea unor entităţi de bază cum ar fi fonemele. Acestea ar putea fi apoi grupate astfel încât să alcătuiască silabe sau cuvinte. Fenomenul coarticulaţiei face, însă, ca două foneme succesive să se influenţeze reciproc, iar rezultatul este o variaţie de la pronunţia standard prin modificarea spectrului de frecvenţe specific fiecărui fonem, prin modificarea amplitudinii sau a duratei. Coarticularea induce astfel un element de confuzie şi limita dintre foneme nu mai este foarte clară. Problema este delicată şi poate influenţa negativ performanţele întregului sistem de recunoaştere a formelor.

Frontiera între extragerea caracteristicilor şi recunoaştere nu este întotdeauna foarte clară [17]. Un modul ideal de extragere a caracteristicilor ar trebui să însemne o recunoaştere foarte uşoară, în timp ce un sistem foarte puternic de recunoaştere nu ar trebui să aibă nevoie de un modul de extragere a caracteristicilor foarte complex. Din punct de vedere practic, însă, este foarte dificil de construit module omnipotente.

Rolul modulului de extragere a caracteristicilor este de a genera un set de însuşiri ale căror valori sunt similare obiectelor care fac parte din aceeaşi categorie şi foarte diferite pentru obiectele din categorii diferite. Această afirmaţie ne conduce la ideea căutării trăsăturilor distinctive care sunt invariante la transformări irelevante asupra intrărilor. În recunoaşterea vorbirii, dorim, de exemplu, ca însuşirile să fie invariante la viteza de pronunţie a cuvintelor de aceeaşi persoană sau de persoane diferite.

Extragerea trăsăturilor, ca şi segmentarea, este o problemă dependentă de tipul aplicaţiei şi implică diverse cunoştinţe specifice domeniului său. În urma acestei etape, se introduce un nivel de abstractizare a intrărilor şi clasificatorul va deveni mai puţin dependent de aplicaţie.

Ca parte a întregului sistem, modulul de recunoaştere primeşte ca intrări vectori pe care îi asignează unei categorii sau unei clase. Implementarea perfectă a recunoaşterii este, în majoritatea cazurilor, imposibilă, de aceea se preferă stabilirea unei marje de eroare în limitele căreia aceasta este considerată corectă. Dificultatea implementării poate să provină din complexitatea intrinsecă a problemei şi atunci decizia este una netrivială.

Procesul prin care clasificatorul înmagazinează informaţia de care are nevoie pentru a fi capabil să asocieze o formă unei categorii se numeşte instruire. Instruirea presupune un model, un set de proceduri şi date cu ajutorul cărora se stabilesc parametrii clasificatorului. Deşi un sistem complex poate oferi o clasificare perfectă a datelor de instruire, se poate constata uneori că performanţele nu mai sunt aceleaşi pentru forme noi. De aceea, un clasificator bun va realiza întotdeauna echilibrul între un model nu prea simplu, care să poată face

distincţia între clase şi unul nu foarte complex, pentru a evita clasificarea slabă a formelor noi.

1.2 Recunoaşterea vorbirii

Vorbirea este un proces care, pentru noi, este foarte natural şi uşor. Încă de la vârste foarte mici căpătăm abilităţile necesare înţelegerii şi transmiterii mesajelor vorbite. Învăţarea se desfăşoară în mai multe etape. Mai întâi recepţionăm mesajul pe care apoi încercăm să îl reproducem. Înţelegerea sensului mesajului nu ne interesează în acest studiu. Pentru o învăţare performantă şi facilă este necesară existenţa unui set de obişnuinţe preliminare, cum sunt structurile elementare de limbă şi modul lor de pronunţare. Se presupune aşadar că persoana este vorbitoare a limbii respective. În caz contrar, învăţarea este ceva mai dificilă pentru că lipsesc cunoştinţele preliminare. În ambele situaţii, însă, o fiinţă umană va fi capabilă să repete cu o mare acurateţe noţiunile pe care şi le-a însuşit în urma unui proces de învăţare.

Tractul vocal şi articulator sunt organe cu proprietăţi neliniare, iar comportamentul lor este afectat de diferiţi factori, de la genul persoanei până la starea ei emoţională. De aceea, felul în care sunt pronunţate cuvintele poate varia din punct de vedere al accentului, al articulării cuvintelor, al volumului sau al vitezei. În fine, pot interveni factorii perturbatori externi cum sunt zgomotul de fond sau ecoul, dar şi caracteristicile electrice ale echipamentelor electronice care sunt folosite drept suport de stocare sau ale liniilor telefonice utilizate în transmisiile la distanţă.

Înţelegerea vorbirii care pentru noi este o obişnuinţă, devine o problemă foarte complexă dacă dorim să o repetăm prin tehnici care ţin de învăţarea şi recunoaşterea automată. Aparent simplă, crearea unui sistem artificial care să recunoască limbajul vorbit s-a dovedit a fi, totuşi, destul de dificilă.

Sistemele actuale de recunoaştere a vorbirii aparţin următoarelor categorii [16]:

a) Sistemele care acceptă un vocabular de dimensiuni reduse, cu 10-100 de cuvinte;

b) Sistemele care pot recunoaşte cuvintele pronunţate izolat unul de celălalt şi pentru care dimensiunea vocabularului poate atinge 10000 de cuvinte;

c) Sisteme care acceptă la intrare fraze, recunoscând cuvintele chiar dacă acestea sunt pronunţate cursiv, fără pauze. Majoritatea acestor sisteme

au, în mod obişnuit, un vocabular restricţionat la un anumit domeniu şi recunosc 1000-5000 de cuvinte.

Majoritatea sistemelor actuale utilizate în aplicaţiile practice fie folosesc un vocabular de dimensiuni reduse, fie recunosc cuvinte pronunţate izolat. Sistemele folosite pentru o comunicare naturală între om şi calculator rămân un deziderat care nu a fost încă atins. Nici o maşină nu a reuşit deocamdată să atingă performanţe maxime în aplicaţiile practice.

Un sistem de recunoaştere a vorbirii va conduce întotdeauna la rezultate cu atât mai bune cu cât va fi folosit de mai puţine persoane. Dacă persoanele care au antrenat sistemul sunt şi cele care îl folosesc, atunci rezultatele sunt şi mai bune. Pe de altă parte, utilizatorii care pronunţă cuvintele cu claritate şi mai rar se fac mai uşor înţeleşi de un astfel de sistem. În contrast cu aceste consideraţii, aplicaţiile de maximă utilitate şi cu caracter universal care ar fi capabile să răspundă unor cerinţe reale şi complexe nu trebuie să folosească vocabular limitat, să impună o anumită modalitate de pronunţie a cuvintelor sau o anumită topică a frazelor, să limiteze numărul persoanelor care pot folosi sistemul ori să fie influenţat de perturbaţiile mediului în care este folosit. Mai mult, un sistem ideal de recunoaştere a vorbirii ar trebui să fie capabil să se adapteze noilor tendinţe ale limbii, să poată înmagazina şi folosi informaţia legată de modificările lexicale, sintactice sau semantice. Toate acestea demonstrează complexitatea domeniului şi dificultatea problemelor care se pun.

Parametrii vorbirii se obţin prin diverse metode de analiză [16], [72] care concentrează caracteristicile semnalului vocal. În felul acesta, se produc secvenţe de vectori de trăsături care, în mod uzual, se numesc *pattern-uri ale vorbirii*. O abordare în recunoaşterea formelor este cea conexionistă, care încorporează cunoştinţele legate de trăsături într-o reţea densă de noduri şi conexiuni, sub forma modelului de calcul cunoscut prin numele de *reţea neurală*. El combină elemente acustico-fonetice, elemente de recunoaşterea formelor şi de inteligenţă artificială.

Abordarea *acustico-fonetică* se bazează pe teoria conform căreia limba vorbită este alcătuită din unităţi fonetice distincte. Acestea sunt caracterizate de un set de proprietăţi care se manifestă în semnalul vorbirii sau în spectru. Deşi aceste proprietăţi acustice ale unităţilor fonetice variază destul de mult în funcţie de contextul în care se află şi de vorbitor, se presupune că există un set de trăsături de bază care pot fi învăţate şi aplicate în situaţii paractice. Primul pas în recunoaşterea vorbirii prin prisma abordării acustico-fonetice este faza de segmentare şi etichetare. Semnalul vocal este separat în regiuni distincte în care proprietăţile acustice ale semnalului sunt reprezentative pentru una dintre

unităţile fonetice sau una dintre clasele de unităţi fonetice. Fiecarui astfel de segment i se ataşează apoi o etichetă care marchează apartenenţa sa la o unitate fonetică sau clasă de unităţi fonetice. Cea de-a doua etapă constă în determinarea în interiorul unui cuvânt sau a unei secvenţe de cuvinte a unităţilor fonetice etichetate la pasul anterior.

Abordarea recunoaşterii vorbirii care se bazează pe *recunoaşterea formelor* este una prin care vectorii de test, a căror apartenenţă la o anumită clasă nu este cunoscută, sunt comparaţi cu un set de vectori de referinţă. Decizia se ia pe baza unei măsuri de similaritate dintre vectorul necunoscut şi fiecare vector de referinţă. Această metodă constă din doi paşi: instruirea vectorilor vorbirii şi recunoaşterea unor vectori noi prin metoda comparaţiei. Cu cât sunt folosite în setul de instruire mai multe versiuni ale formei care trebuie recunoscută, cu atât mai performantă va fi procedura de instruire în extragerea proprietăţilor acustice ale sale.

Abordarea prin prisma *inteligenţei artificiale* a recunoaşterii vorbirii este o metodă hibridă şi exploatează idei şi concepte din cele două abordări descrise mai sus, cea acustico-fonetică şi cea bazată pe recunoaşterea formelor. Ideea fundamentală este aceea de a încorpora în sistemul de recunoaştere cunoştinţe care provin de la o varietate largă de surse. O tehnică ce poate fi încadrată în această clasă este folosirea unui sistem expert pentru segmentare şi etichetare care realizează această operaţie nu numai pe baza informaţiilor acustice, aşa cum se întâmplă prin metodele acustico-fonetice. Un sistem expert poate integra informaţii fonetice, lexicale, sintactice sau semantice dar, în plus, este capabil să înveţe şi să se adapteze.

Folosirea reţelelor neurale reprezintă o abordare distinctă a recunoaşterii vorbirii, dar poate fi privită şi ca o structură care combină caracteristici ale tehnicilor descrise mai sus.

Recunoaşterea vocalelor

Producţia vocală este o înşiruire de unităţi fonetice care au caracteristici distincte şi care stau la baza comunicării mesajelor între oameni. Unităţile fonetice (fonemele) au caracteristici fonice şi articulatorii proprii care ajută la înţelegerea lor şi care fac posibilă transmiterea mesajelor verbale. Modul lor de pronunţie este, însă, influenţat de caracteristicile fizice ale aparatului fonator propriu fiecărui individ. Aceste variaţii de pronunţie nu influenţează calitatea informaţiei transmise între doi parteneri de dialog, însă creează probleme la recunoaşterea automată a fonemelor. Unităţile fonetice alcătuiesc sistemul fonetic al unei limbi şi sunt împărţite în două categorii: vocale şi

consoane. Vocalele reprezintă sunete produse la trecerea liberă a coloanei de aer prin traiectul vocal, iar consoanele apar atunci când fluxul de aer întâlneşte diferite obstacole. Vocalele sunt produse de vibraţia coardelor vocale, dar şi de buze, poziţia limbii, a dinţilor şi a altor organe vocale care influenţează sunetul rezultat. Ele au în general o durată mai mare în comparaţie cu consoanele şi sunt mai bine definite în domeniul spectral.

Sistemele de recunoaştere a vorbirii folosesc recunoaşterea vocalelor ca o modalitate de creştere semnificativă a abilităţii lor de recunoaştere.

Recunoaşterea cuvintelor izolate

Despre sistemele care recunosc entităţi discrete, de regulă cuvinte, se spune că folosesc la recunoaşterea cuvintelor izolate. În faza de recunoaştere, se presupune că vorbitorii utilizează în mod deliberat cuvinte rostite rar, cu pauze între ele. În mod uzual, acestea au minim 200 ms pentru ca sistemul să nu le confunde cu fonemele care în general sunt mai scurte. Faptul că limita dintre cuvinte este uşor de stabilit simplifică semnificativ recunoaşterea. Limitele pot fi localizate prin diverse tehnici cum ar fi folosirea unor algoritmi pentru detecţia punctelor de start şi de sfârşit ale cuvintelor. Acest tip de sistem de recunoaştere este util aplicaţiilor în care se utilizează comenzi sub forma unor cuvinte simple, dintr-un vocabular redus.

Recunoaşterea vorbitorilor

Expresia vocală este o caracteristică proprie fiecărui individ. Din acest motiv, este posibilă recunoaşterea în condiţii normale a unui vorbitor, de exemplu a partenerului unei conversaţii telefonice. Variaţiile individuale între vorbitori sunt influenţate de doi factori. În primul rând, este vorba de proprietăţile fizice ale aparatului fonator care determină caracteristicile de frecvenţa ale semnalului vocal. Pe de altă parte, o frază nu este pronunţată la fel de doi vorbitori. Întotdeauna vor apărea diferenţe în debitul verbal, diferenţe de ton, de accent etc. Aşa cum sugerează şi termenul, *recunoaşterea unui vorbitor* se referă la problema identificării unui individ dintr-o populaţie, fără a interesa mesajul transmis.

Atunci când recunoaşterea vorbitorilor se face pe baza unui vocabular limitat, bine determinat, sau pe baza unui text impus, este vorba de tehnica numită *fixed-text speaker recognition*. Dacă recunoaşterea este independentă de text, este vorba de *free-text speaker recognition*.

În strânsă legătură cu recunoaşterea vorbitorului este *verificarea vorbitorului*. De această data problema constă în determinarea identităţii declarate de

un individ. De regulă, se compară distanţa dintre expresia sa vocală şi referinţa sa personală.

1.3 Structura lucrării

Teza este organizată în şapte capitole care conţin părţi teoretice şi descrieri ale experimentelor de recunoaştere a formelor, iar contribuţiile originale sunt argumentate în detaliu. Experimentele de recunoaştere vocală sunt diverse, pornind de la recunoaşterea cuvintelor izolate, continuând cu recunoaşterea vorbitorilor şi terminând cu recunoaşterea vocalelor. Bazele de date folosite acoperă o gamă largă, unele fiind publice, altele alcătuite special pentru experimentele din această lucrare. În unele experimente am utilizat şi baze de date care nu sunt destinate strict recunoaşterii vorbirii, demonstrând astfel că algoritmii propuşi pot fi folosiţi şi pentru probleme generale de recunoaştere a formelor.

Capitolul 2 face o prezentare a tipurilor fundamentale de reţele neurale şi a tehnicilor de instruire care stau la baza modelelor introduse în celelalte capitole. Neuronul McCulloch-Pitts reprezintă punctul de pornire al domeniului conexionist. Deşi este foarte simplu ca structură, acest neuron artificial inspirat ca structură şi funcţionare de procesele neurale care au loc în creierul uman s-a dovedit a avea un mare potenţial de calcul. Pot fi implementate cu ajutorul său chiar şi funcţii logice oricât de complicate. Perceptronul introdus de Rosenblatt este un model mai evoluat, fiind capabil să înveţe să clasifice vectori liniar separabili. Perceptronul multistrat constă dintr-o colecţie de perceptroni elementari, organizaţi ierarhic. Este un model adaptiv avansat care poate fi instruit şi care poate implementa funcţii oricât de complicate. Datorită acestei proprietăţi, poate fi utilizat ca şi clasificator pentru vectori neseparabili liniar. Algoritmul de instruire *backpropagation of error* se bazează pe tehnica gradientului negativ şi permite ajustarea iterativă, supervizată parametrilor perceptronului multistrat pornind de la un set de date de instruire. Reţeaua neurală cu întârziere în timp este un perceptron multistrat extins pentru probleme de recunoaştere a vorbirii prin introducerea unor elemente de întârziere a semnalelor de intrare. Învăţarea competitivă constă dintr-un proces de selecţie în urma căruia vectorii de instruire sunt grupaţi după caracteristicile comune. Algoritmul *self-organizing feature map* este nesupervizat şi realizează o transformare a unui hiperspaţiu cu un număr mare de dimensiuni într-unul unidimensional sau bidimensional, iar reprezentarea în plan a acestei transformări se numeşte hartă cu autoorganizare. Algoritmii din clasa *learning vector quantization* sunt supervizaţi şi aproximează setul de date de instruire printr-o mulţime finită de

vectori numiţi prototipuri.

În capitolul 3 este prezentat un set de experimente de recunoaştere a cuvintelor izolate şi a vorbitorilor folosind trei modele clasice de reţele neurale: perceptronul multistrat, reţeaua neurală cu întârziere în timp şi harta cu autoorganizare. Cele două baze de date, Speechdata şi Telephdata, pe care le-am folosit în aceste teste constau din cuvinte izolate pronunţate de diverşi vorbitori mai întâi în condiţii de studio, fără perturbaţii, şi apoi prin telefon. Pentru preprocesarea semnalelor vocale se foloseşte o cunoscută metodă de calcul al coeficienţilor cepstrali. Problema instruirii şi cea a recunoaşterii informaţiei din aceste două baze de date sunt deosebit de complicate şi rezultatele experimentale demonstrează acest lucru. Recunoaşterea cuvintelor izolate conduce la rezultate satisfăcătoare în majoritatea cazurilor, însă recunoaşterea vorbitorilor ridică probleme deosebite.

Reţelele neurale concurente introduse în capitolul 4 reprezintă un model conexionist de recunoaştere care porneşte de la observaţia că o reţea neurală este mai eficientă atunci când implementează o problemă mai puţin complexe sau când este specializată pe o subproblemă a problemei globale de recunoaştere. Modelul propus în acest capitol este o colecţie de reţele neurale care lucrează în paralel, iar decizia în recunoaştere se ia conform regulii *câştigătorul ia tot*. Fiecare reţea neurală componentă este instruită individual cu un set propriu de date. Fiind vorba de module care rezolvă probleme cu o complexitate intrinsecă mai scăzută, numărul de neuroni pentru fiecare modul poate fi redus semnificativ. Am instruit reţele neurale concurente având drept unităţi elementare perceptroni multistrat, reţele neurale cu întârziere în timp şi hărţi cu autoorganizare pentru aceeaşi problemă de recunoaştere a vorbitorilor din capitolul 3, iar performanţele pentru cele două baze de date au crescut semnificativ. Cu acelaşi model am implementat şi recunoaşterea unui subset de şase vocale din limba română, baza de date fiind alcătuită prin segmentarea manuală a semnalelor vocale, în scopul extragerii secvenţelor vocalizate. Preprocesarea constă din calculul coeficienţilor cepstrali care sunt vectorii de instruire. Reţeaua neurală concurentă care foloseşte hărţi cu autoorganizare specializate pe recunoaşterea câte unei vocale are performanţe mai bune decât o hartă unică, instruită cu toate vocalele.

În capitolul 5, problema recunoaşterii este abordată dintr-o perspectivă diferită pentru că ne punem problema creşterii performanţelor prin selectarea acelor caracteristici ale vectorilor de instruire şi de test care contribuie cel mai mult la o recunoaştere corectă. Fiecărei componente a vectorilor de intrare i se asociază câte un coeficient care reprezintă importanţa acestuia, iar coeficienţii sunt instruiţi în aşa fel încât să minimizeze eroarea de cuantizare a

unei reţele neurale de tip LVQ. Algoritmul RLVQ calculează printr-o tehnică euristică aceşti coeficienţi. Operatorii OWA sunt o familie de operatori de agregare care se bazează pe reordonarea criteriilor care trebuie satisfăcute într-un proces de clasificare. Algoritmul OWA-RLVQ propus de noi calculează dinamic relevanţele intrărilor ca ponderi ale unui operator OWA, făcând totodată o legătură importantă între algoritmul RLVQ şi modelul matematic al operatorilor OWA. Performanţele algoritmului nostru sunt comparate cu ratele de recunoaştere obţinute cu LVQ şi RLVQ şi rezultatele înregistrate de OWA-RLVQ sunt superioare pentru toate cele trei baze de date utilizate în teste. Dovedim astfel că metoda noastră este competitivă atât pentru probleme de recunoaştere vocală cât şi pentru probleme generale de recunoaştere a formelor.

Capitolul 6 introduce două noi metode de calcul al relevanţelor pentru componentele vectorilor de intrare cu ajutorul informaţiei mutuale pe de o parte şi a energiei informaţionale pe de altă parte. Estimarea pe baza unor date de instruire a informaţiei mutuale definită cu ajutorul entropiei Shannon ridică probleme computaţionale serioase. Se pot folosi metode alternative cum ar fi cea care calculează entropia Renyi pornind de la estimarea densităţilor de probabilitate cu ferestre Parzen. Informaţia mutuală pătratică dintre două variabile aleatoare obţinută fie prin similaritatea cu entropia Renyi pătratică, fie pe baza inegalităţii lui Pinsker devine un criteriu care poate fi maximizat iterativ într-o problemă de gradient pozitiv. Algoritmul MIRLVQ propus de noi calculează iterativ relevanţele vectorilor de intrare maximizând informaţia mutuală pătratică dintre o variabilă aleatoare continuă ale cărei realizări sunt calculate pe baza datelor de instruire şi o variabilă aleatoare discretă ale cărei realizări sunt clasele din care fac parte vectorii de instruire. Energia informaţională este o mărime pe baza căreia se calculează măsura $o(X, Y)$, unde X şi Y sunt două variabile aleatoare, care este similară informaţiei mutuale. Al doilea algoritm propus de noi, numit ERLVQ, calculează relevanţele componentelor din vectorii de intrare prin maximizarea măsurii $o(X, Y)$ şi obţinem o formulă de actualizare similară celei din MIRLVQ. Experimentele realizate cu aceleaşi baze de date folosite în capitolul 5 arată că noua metodă este superioară din punct de vedere al ratei de recunoaştere algoritmului OWA-RLVQ.

Capitolul 7 care este o sinteză prezintă contribuţiile pe care teza le aduce la dezvoltarea domeniului recunoaşterii formelor şi posibilele direcţii viitoare de dezvoltare a rezultatelor obţinute în lucrare.

Capitolul 2

Reţele neurale. Concepte fundamentale

Reţelele neurale constau din elemente de procesare numite neuroni care pot fi organizate în structuri regulate, pe nivele ierarhice. Adaptarea parametrilor reţelelor neurale la datele de intrare se poate face printr-un proces de instruire. Acest capitol prezintă arhitecturile neurale fundamentale pe care le-am folosit la dezvoltarea modelelor originale propuse în teză.

2.1 Introducere

Reţelele neurale se pot defini, pe de o parte, ca o clasă de algoritmi matematici deoarece o reţea poate fi privită ca o reprezentare grafică pentru o largă categorie de algoritmi. Pe de altă parte, reţelele neurale imită reţelele neurale biologice proprii organismelor vii. Privite prin prisma cunoştinţelor limitate pe care le avem despre reţelele neurale biologice, cea mai potrivită definiţie se apropie de cea algoritmică.

Deşi inspirate din biologie, există mari diferenţe între reţelele neurale artificiale şi cele naturale, neexistând încă modele care să concureze cu succes performanţele creierului uman. Calculul neural se realizează pe o reţea densă de noduri şi conexiuni. Nodurile lucrează în mod colectiv, în paralel, şi se numesc *neuroni artificiali* sau, simplu, *neuroni*. Aceştia sunt configuraţi în arhitecturi regulate, pot fi organizaţi pe nivele ierarhice şi sunt permise conexiuni în cadrul aceluiaşi nivel sau către nivele adiacente. Puterea unei conexiuni se exprimă printr-o valoare numerică numită *pondere* care poate fi modificată. În acest capitol vom vedea care sunt principalele modele neurale şi care sunt cele mai importante metode folosite la instruirea neuronilor şi a reţelelor de neuroni.

2.2 Modelul McCulloch-Pitts

Rezultatele primelor cercetări referitoare la calculul inspirat de procesele naturale din creierul uman au fost publicate în anul 1943 de către Warren S. McCulloch şi Walter Pitts [55]. Ei au pornit de la ideea că activitatea nervoasă are un caracter de tipul *totul sau nimic*, ceea ce înseamnă că procesele cerebrale ar putea fi descrise pe baza calculului logic. Aceste consideraţii au condus la crearea primului model neural care acceptă la intrările $x_i, i = 1, \ldots, n$ semnale 0 sau 1, iar ieşirea y se calculează pe baza regulii

$$y = \begin{cases} 1, & \text{dacă } (\sum_{i=1}^{n} w_i\, x_i) \geqslant T \\ 0, & \text{dacă } (\sum_{i=1}^{n} w_i\, x_i) < T \end{cases} . \tag{2.1}$$

Mărimile $w_i, i = 1, \ldots, n$ se numesc *ponderi sinaptice* şi au valori 1 sau -1, iar T se numeşte *prag al neuronului*.

Modelul general al neuronului artificial este cel din figura 2.1.

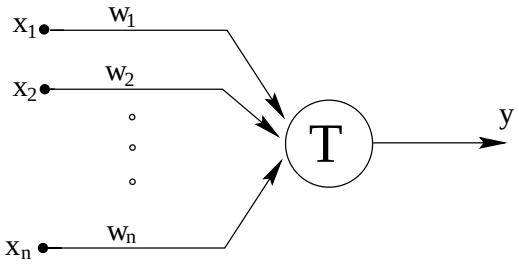

Figura 2.1: Modelul neuronului artificial.

Deşi este foarte simplu ca structură, potenţialul de calcul al neuronului artificial este mare. Folosind scheme cu mai mulţi neuroni şi alegând valorile potrivite pentru ponderi şi praguri, se pot implementa de la operaţiile logice fundamentale până la funcţiile logice oricât de complicate. Deoarece orice funcţie booleană poate fi redusă la una dintre formele canonice conjunctivă sau disjunctivă, ea poate fi implementată de operaţiile NOT, AND şi OR. Aceste operaţii elementare sunt implementate de modelele din figura 2.2 [97].

Folosind neuronul McCulloch-Pitts şi proprietatea sa de întârziere a semnalului cu o unitate de timp, se pot construi circuite digitale secvenţiale. În aceste condiţii, o buclă dinspre ieşirea neuronului înspre intrarea sa va crea imediat modelul unei celule de memorie, ca în figura 2.3 [97].

Am arătat cum se pot implementa porţile NOT, AND, OR şi celula de memorie, dar scopul neuronilor artificiali nu este cel de a reconstrui elementele

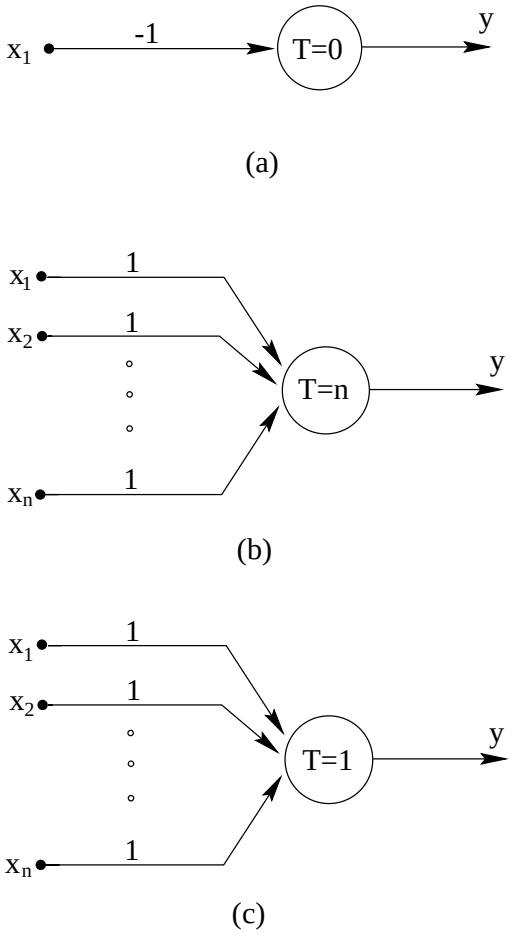

Figura 2.2: Funcţii logice implementate folosind neuronul McCulloch-Pitts: (a)NOT, (b)AND, (c)OR.

esenţiale ale unui sistem digital. Vom vedea, însă, cum se poate exploata puterea de calcul a modelului neural şi cum se poate extinde acesta pentru a construi entităţi ce pot fi supuse unui proces de învăţare.

Deşi oferă o metodă elegantă de implementare a funcţiilor logice şi este caracterizat de definiţia sa foarte precisă, neuronul propus de McCulloch şi Pitts face o serie de simplificări care îl îndepărtează de modelul biologic. El operează într-un spaţiu discret deoarece permite doar intrări şi ieşiri din domeniul $\{0, 1\}$, dar şi în timp discret când neuronii dintr-o reţea sunt sincronizaţi. Ponderile şi pragurile sale sunt, de asemenea, fixate şi sunt stabilite în faza proiectării.

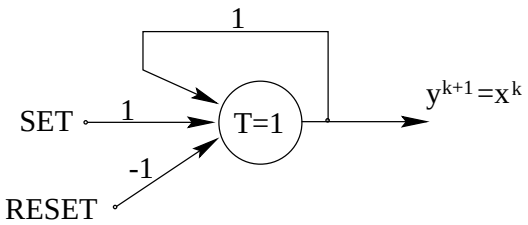

Figura 2.3: Celula de memorie implementată cu neuronul McCulloch-Pitts.

2.3 Perceptronul

Perceptronul [97] introdus de Frank Rosenblatt în 1958 a fost prima maşină capabilă să înveţe. El reprezintă cea mai simplă formă de reţea neurală care poate clasifica vectori liniar separabili, adică vectori care, atunci când sunt reprezentaţi într-un hiperspaţiu, pot fi separaţi cu un hiperplan. Adaptarea perceptronului constă în ajustarea ponderilor sinaptice şi a pragurilor în aşa fel încât, în final, să clasifice corect vectorii din mulţimea de instruire.

În principiu, operaţiile pe care le realizează perceptronul sunt cele pe care le implementează neuronul McCulloch-Pitts. Aşadar, un perceptron este un element care realizează o combinaţie liniară a intrărilor, urmat de un element de procesare care calculează valoarea ieşirii.

Operaţia de bază implementată de un perceptron este:

$$y = f\left(\sum_{i=1}^{n} w_i\, x_i - \theta\right).$$

În relaţia de mai sus, f se numeşte *funcţie de activare a perceptronului*, iar θ este *pragul perceptronului*. Domeniul de definiţie al funcţiei f este mulţimea valorilor activării funcţiei f, $\sum_{i=1}^{n} w_i\, x_i$. Această sumă se poate rescrie prin produsul scalar $\mathbf{w}^t\,\mathbf{x}$. Am notat prin

$$\mathbf{w} = [w_1\; w_2 \ldots w_n]^t$$

şi

$$\mathbf{x} = [x_1\; x_2 \ldots x_n]^t$$

valorile ponderilor, respectiv ale intrărilor perceptronului.

În practică una dintre intrări este fixă, având de regulă valoarea -1, iar ponderea corespunzătoare ei este pragul perceptronului.

Dacă funcţia de activare este *signum*:

$$f(\mathbf{w}^t\,\mathbf{x}) = \mathrm{sgn}(\mathbf{w}^t\,\mathbf{x}) = \begin{cases} -1, & \text{dacă } \mathbf{w}^t\,\mathbf{x} < 0 \\ 1, & \text{dacă } \mathbf{w}^t\,\mathbf{x} \geqslant 0 \end{cases}, \tag{2.2}$$

atunci acest model va implementa un perceptron discret. Dacă funcţia de activare este continuă:

$$f(\mathbf{w}^t\,\mathbf{x}) = \frac{2}{1+e^{-\lambda\,\mathbf{w}^t\,\mathbf{x}}} - 1, \tag{2.3}$$

atunci se implementează un perceptron continuu. În relaţia (2.3), λ se numeşte *câştigul perceptronului.*

Pentru că funcţiile de activare descrise de relaţiile (2.2) şi (2.3) iau valori în intervalul $[-1,1]$, adică atât valori pozitive cât şi valori negative, cele două modele se vor numi *perceptron bipolar discret* şi *perceptron bipolar continuu.* La implementarea unui *perceptron unipolar discret* şi a unui *perceptron unipolar continuu*, care au valori de ieşire în intervalul $[0,1]$, putem folosi funcţiile:

$$f(\mathbf{w}^t\,\mathbf{x}) = \begin{cases} 0, & \text{dacă } \mathbf{w}^t\,\mathbf{x} < 0 \\ 1, & \text{dacă } \mathbf{w}^t\,\mathbf{x} \geqslant 0 \end{cases} \tag{2.4}$$

şi

$$f(\mathbf{w}^t\,\mathbf{x}) = \frac{1}{1+e^{-\lambda\,\mathbf{w}^t\,\mathbf{x}}}. \tag{2.5}$$

Cu scopul de a descrie procesul de învăţare a unui perceptron discret, să presupunem că dispunem de două mulţimi de vectori liniar separabili care aparţin claselor X_1 şi X_2. Prin reuniunea celor două mulţimi, alcătuim întregul set de vectori. Pentru a implementa corect această separare liniară, perceptronul va trebui să aibă un set de ponderi $\mathbf{w}$ pentru care

$$\begin{cases} \mathbf{w}^t\,\mathbf{x} < 0, & \text{când } \mathbf{x} \in X_1 \\ \mathbf{w}^t\,\mathbf{x} \geqslant 0, & \text{când } \mathbf{x} \in X_2. \end{cases}$$

Prin instruirea perceptronului putem găsi, în condiţiile teoremei de convergenţă a perceptronului discret enunţată mai jos, un $\mathbf{w}$ care îndeplineşte relaţia anterioară pentru orice vector $\mathbf{x}$. Pentru aceasta, la instruirea unui perceptron ca şi clasificator liniar trebuie să aplicăm iterativ următorii paşi fiecărui vector $i = 1, \ldots, n$:

Pasul 1. *Dacă vectorul $\mathbf{x}_i$ este clasificat corect, atunci nu se fac corecţii asupra lui $\mathbf{w}$.*

Pasul 2. *Dacă vectorul $\mathbf{x}_i$ nu este clasificat corect, atunci*

$$\begin{cases} \mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta\,\mathbf{x}_i, & \textit{dacă } \mathbf{w}^{(t)t}\,\mathbf{x}_i < 0 \textit{ şi } \mathbf{x}_i \in X_1 \\ \mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta\,\mathbf{x}_i, & \textit{dacă } \mathbf{w}^{(t)t}\,\mathbf{x}_i \geqslant 0 \textit{ şi } \mathbf{x}_i \in X_2. \end{cases}$$

În aceste relaţii, t este pasul de instruire, η este un parametru ajustabil care se numeşte *rată de învăţare*, iar $\mathbf{w}^{(t+1)}$ este vectorul de ponderi care rezultă în urma ajustării la pasul $t+1$.

Teorema de convergenţă a perceptronului discret cu un increment fixat arată că instruirea cu o mulţime liniar separabilă de vectori se realizează într-un număr finit de paşi.

Teorema de convergenţă a perceptronului discret. *Fie submulţimile de vectori X_1 şi X_2 liniar separabile. Instruirea unui perceptron discret cu aceşti vectori are loc într-un număr t_0 de iteraţii, adică*

$$\mathbf{w}^{(t_0)} = \mathbf{w}^{(t_0+1)} = \dots$$

este vectorul soluţie.

2.4 Reguli de învăţare

Datele sunt stocate într-o reţea neurală printr-un *proces de învăţare*. Acesta constă dintr-o recalculare a parametrilor reţelei, forţându-se un anumit răspuns pentru o intrare specifică. Un răspuns particular poate să genereze sau nu o corecţie.

Învăţare supervizată şi învăţare nesupervizată

Procesul de învăţare se bazează pe conceptul de *feedback*. Acesta conduce la introducerea unei relaţii între intrarea şi ieşirea sistemului. Există două tipuri de învăţare: cea cu supervizare şi cea fără supervizare.

În învăţarea supervizată presupunem că fiecare ieşire a reţelei este comparată cu o ieşire dorită d asociată vectorului introdus. Diferenţele dintre răspunsul dorit şi cel generat de reţea servesc drept măsură a erorii care este folosită pentru a corecta parametrii reţelei.

În învăţarea fără supervizare, răspunsul dorit nu se cunoaşte şi din acest motiv nu se poate stabili o măsură a erorii în funcţie de care să se corecteze comportarea reţelei. De aceea, mecanismul de învăţare trebuie să prevadă o modalitate de ajustare a parametrilor pe baza similitudinilor între vectori. Reţeaua trebuie să descopere singură proprietăţi, regularităţi ale vectorilor care să conducă la separare. Această modalitate de adaptare se numeşte *autoorganizare*.

Regula generală de învăţare

Un neuron este un element adaptiv care îşi modifică ponderile în funcţie de intrare, de ieşire şi de răspunsul dorit pentru fiecare vector pe baza regulii de instruire utilizată. Atunci când nu există răspuns dorit, ponderile se modifică în funcţie doar de intrare şi de ieşire, deci nesupervizat.

În studiul reţelelor neurale a fost adoptată o regulă generală de instruire [97] care, prin diferite particularizări, produce o serie de noi reguli. Pentru un neuron ca cel din figura 2.1, vectorul ponderilor $\mathbf{w} = [w_1\, w_2 \ldots w_n]^t$ se modifică proporţional cu produsul dintre intrările $\mathbf{x}$ şi *semnalul de învăţare* s. Semnalul de învăţare se calculează ca o funcţie de vectorul $\mathbf{w}$, de intrarea $\mathbf{x}$ şi, atunci când este cazul, de ieşirea dorită d. Odată făcute aceste notaţii, regula de actualizare a ponderilor neuronului se scrie astfel:

$$\mathbf{w}(t+1) = \eta s(\mathbf{w}(t), \mathbf{x}(t), d(t))\mathbf{x}(t) \tag{2.6}$$

pentru cazul continuu la momentul $t+1$ şi sub forma:

$$\mathbf{w}^{(t+1)} = \eta s(\mathbf{w}^{(t)}, \mathbf{x}^{(t)}, d^{(t)})\mathbf{x}^{(t)} \tag{2.7}$$

în cazul discret la pasul $t+1$.

Principalele reguli de instruire a neuronilor se obţin prin particularizarea regulii generale.

Regula lui Hebb

Această regulă este nesupervizată şi implementează *postulatul învăţării* formulat de Hebb în cartea sa apărută în 1949 [31] care propune ideea învăţării asociative la nivel celular:

> *Dacă celula A excită în mod repetat celula B facând-o să genereze un impuls, atunci are loc un proces de creştere (schimbare metabolică) într-una sau ambele celule, astfel încât eficienţa excitării lui B de către A creşte.*

Cu alte cuvinte, vectorii de intrare cu frecvenţa cea mai mare vor avea influenţa cea mai puternică asupra ponderilor conexiunilor.

Această regulă particularizează semnalul de învăţare astfel:

$$s = f(\mathbf{w}^t\mathbf{x}),$$

adică semnalul de învăţare este chiar ieşirea neuronului. Atunci:

$$\Delta\mathbf{w} = \eta f(\mathbf{w}^t\mathbf{x})\mathbf{x},$$

unde η este o constantă de învăţare.

Regula perceptronului

Aşa cum a fost definită de Rosenblatt, regula de instruire a perceptronului consideră că semnalul de învăţare este diferenţa dintre răspunsul dorit şi cel actual al neuronului:

$$s = d - y. \tag{2.8}$$

Din (2.6) şi (2.8) obţinem formula de ajustare a ponderilor perceptronului:

$$\Delta \mathbf{w} = \eta \left[d - \text{sgn}(\mathbf{w}^t \mathbf{x}) \right] \mathbf{x} \tag{2.9}$$

unde ieşirea se calculează pe baza funcţiei bipolare discrete

$$y = \text{sgn}(\mathbf{w}^t \mathbf{x}).$$

Se observă că această regulă se aplică doar pentru neuroni cu răspuns binar, iar ponderile sunt ajustate doar dacă ieşirea actuală y este incorectă. Această observaţie ne permite să rescriem relaţia (2.9) ţinând cont şi de faptul că ieşirile dorite pot lua doar valorile 1 şi -1 astfel:

$$\Delta \mathbf{w} = \begin{cases} -2\eta \mathbf{x}, & \text{când } d = -1 \text{ şi } \text{sgn}(\mathbf{w}^t \mathbf{x}) = 1 \\ 2\eta \mathbf{x}, & \text{când } d = 1 \text{ şi } \text{sgn}(\mathbf{w}^t \mathbf{x}) = -1 \end{cases}.$$

Regula delta

Regula delta este aplicabilă doar pentru funcţii continue de activare şi învăţare supervizată. Poate fi dedusă căutând varianta de scădere cea mai rapidă a erorii pătratice medii dintre ieşirea actuală y a neuronului şi ieşirea dorită d. Eroarea pătratică medie

$$E = \frac{1}{2}(d - y)^2$$

este echivalentă cu:

$$E = \frac{1}{2} \left[d - f(\mathbf{w}^t \mathbf{x}) \right]^2,$$

iar gradientul ei va fi:

$$\nabla E = -(d - y) f'(\mathbf{w}^t \mathbf{x}) \mathbf{x}. \tag{2.10}$$

Pentru ca algoritmul să aibă eficienţa maximă, ponderile vor fi modificate în sensul gradientului negativ al erorii astfel:

$$\Delta \mathbf{w} = -\eta \nabla E, \tag{2.11}$$

unde η este o constantă pozitivă. Din relaţiile (2.10) şi (2.11) obţinem formula generală de modificare a ponderilor pentru regula de instruire delta:

$$\Delta \mathbf{w} = \eta (d - y) f'(\mathbf{w}^t \mathbf{x}) \mathbf{x}. \tag{2.12}$$

Comparând regula generală de instruire în cazul continuu dată de relaţia (2.6) şi regula delta exprimată prin relaţia (2.12), observăm că în acest caz semnalul de învăţare este:

$$s = \left[d - f(\mathbf{w}^t\mathbf{x})\right] f'(\mathbf{w}^t\mathbf{x}).$$

Regula delta a fost introdusă de McClelland şi Rumelhart în 1986 şi se mai numeşte şi regula de instruire a perceptronului continuu, putând fi extinsă şi pentru instruirea reţelelor neurale multistrat.

Regula *câştigătorul ia tot*

Această regulă se foloseşte pentru instruirea nesupervizată a unor reţele de neuroni, fiind un exemplu de învăţare competitivă. Spre deosebire de regulile prezentate anterior, regula *câştigătorul ia tot* se aplică asupra unei colecţii de p neuroni organizaţi pe un singur strat.

Învăţarea se bazează pe faptul că unul dintre neuroni, de exemplu neuronul m, are răspuns maxim pentru vectorul $\mathbf{x}$, fiind declarat *câştigător*. Câştigătorul este selectat pe baza următorului criteriu:

$$\mathbf{w}_m^t\mathbf{x} = \max_{i=1,2\ldots p} (\mathbf{w}_i^t\mathbf{x}).$$

Drept urmare, doar ponderile

$$\mathbf{w}_m = [w_{m1}\ w_{m2}\ldots w_{mn}]^t$$

care converg către el vor fi actualizate în următorul pas al învăţării. Formula de ajustare a ponderilor este:

$$\Delta\mathbf{w}_m = \alpha(\mathbf{x} - \mathbf{w}_m),$$

unde α este o constantă pozitivă cu valori mici.

De regulă, actualizarea ponderilor se face pentru neuronii dintr-o vecinătate a neuronului câştigător. Ponderile se iniţializează cu valori aleatoare şi sunt normalizate pe parcursul învăţării.

2.5 Modele de reţele neurale artificiale

O reţea neurală poate fi definită ca o interconectare de neuroni cu proprietatea că ieşirile unor neuroni sunt conectate prin intermediul conexiunilor sinaptice cu intrările altor neuroni sau ale lor însele.

Pornind de la această definiţie, se pot imagina diverse arhitecturi de reţele neurale. Cele mai populare sunt reţelele *feedforward* şi reţelele *feedback* [30], [97].

Reţele neurale feedforward

O arhitectură elementară feedforward constă dintr-o colecţie de neuroni care permit transmiterea semnalelor într-un singur sens, dinspre intrări înspre ieşiri.

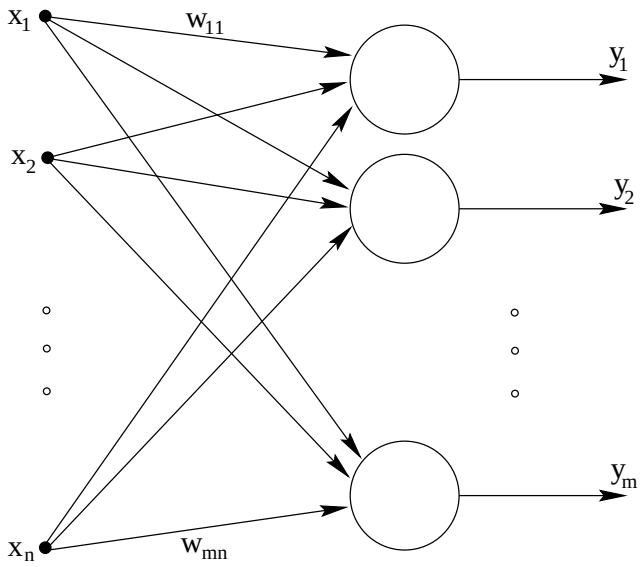

Figura 2.4: Reţea neurală feedforward cu un singur strat de neuroni.

În figura 2.4 am reprezentat o reţea neurală feedforward cu un singur strat de neuroni. Notăm cu $\mathbf{x}$ şi $\mathbf{y}$ vectorii intrărilor şi ieşirilor:

$$\mathbf{x} = [x_1\ x_2 \ldots x_n]^t$$

$$\mathbf{y} = [y_1\ y_2 \ldots y_m]^t.$$

Dacă notăm cu w_{ji} ponderea conexiunii care leagă neuronul j cu intrarea i, atunci matricea $\mathbf{W}$ a ponderilor se va scrie astfel:

$$\mathbf{W} = \begin{bmatrix} \mathbf{w_1} \\ \mathbf{w_2} \\ \vdots \\ \mathbf{w_n} \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \ldots & w_{1n} \\ w_{21} & w_{22} & \ldots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \ldots & w_{mn} \end{bmatrix}.$$

Folosind aceste notaţii, ieşirile neuronilor se vor calcula conform relaţiei:

$$y_i = f(\mathbf{w}_i^t \mathbf{x}_i), \qquad i = 1, \ldots, n.$$

Reţelele neurale feedforward pot fi extinse grupând neuronii pe straturi. Arhitecturile multistrat se obţin prin cascadarea mai multor straturi de neuroni.

Stratul final se numeşte *strat de ieşire*, iar celelalte straturi de neuroni se numesc *straturi ascunse*. Intrările $\mathbf{x}$ şi ieşirile $\mathbf{y}$ se numesc *vectori de intrare* şi *vectori de ieşire*. În timpul instruirii, ieşirile reţelei sunt comparate cu un set de ieşiri dorite şi, în funcţie de diferenţele care apar, sunt modificate valorile ponderilor $\mathbf{W}$.

Reţele neurale feedback

Reţelele neurale feedback se obţin din reţele neurale feedforward prin conectarea ieşirilor reţelei la intrările sale, ca în figura 2.5. Prin buclele care se închid peste reţea, fiecare ieşire va fi controlată de valorile celorlalte ieşiri. Ieşirea de la momentul t controlează ieşirea de la momentrul $t + \Delta$:

$$y_j(t + \Delta) = \sum_{i=1}^{n} w_{ji}\, y_i(t), \quad j = 1, \ldots, n \tag{2.13}$$

unde diferenţa de timp Δ este introdusă de elementele de întârziere plasate pe buclele de feedback. Intrările $x(t)$ sunt folosite pentru iniţializarea reţelei, după care dispar şi sistemul rămâne autonom. Din acest moment, el se autoîntreţine datorită legăturilor dintre ieşiri şi intrări.

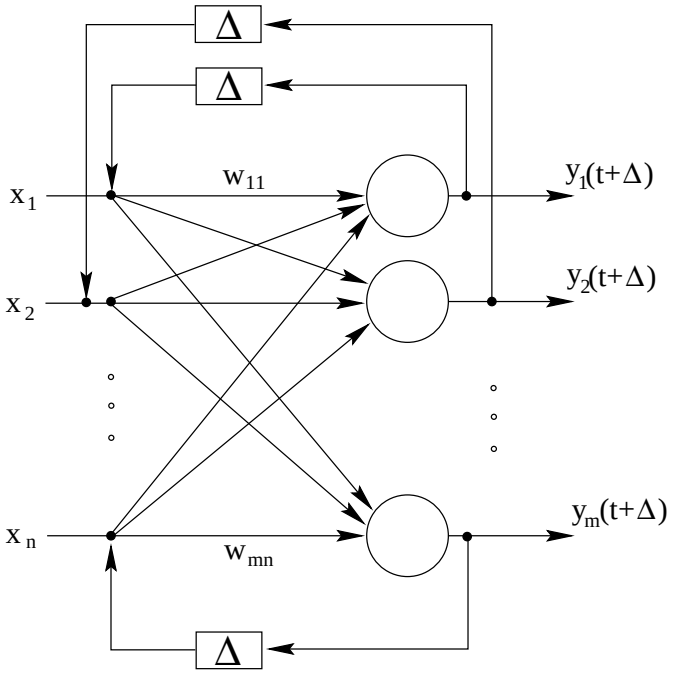

Figura 2.5: Reţea neurală feedback.

În situaţia în care ieşirile sunt înregistrate la intervale determinate de timp, reţeaua se numeşte cu *timp discret*. Din acest motiv, putem rescrie ecuaţia (2.13) asimilând intârzierea Δ cu unitatea şi momentele de timp cu întregi pozitivi astfel:

$$y_j^{(t+1)} = \sum_{i=1}^{n} w_{ji}\, y_i^{(t)}, \quad j = 1, \ldots, n. \tag{2.14}$$

Reţeaua neurală din figura 2.5 se numeşte *recurentă* pentru că răspunsul său la momentul de timp $t+1$ depinde de întregul istoric, începând cu momentul $t = 0$. Relaţia (2.14) arată că pentru a calcula $y_j^{(t+1)}$ trebuie să calculăm mai întâi $y_j^{(t)}$. Pentru aceasta, este nevoie de $y_j^{(t-1)}$ etc.

Reţelele recurente lucrează de obicei cu o reprezentare discretă a datelor, folosind neuroni cu funcţie de activare discretă. Un sistem cu intrări discrete şi o reprezentare discretă a datelor se numeşte *automat*. Reţelele neurale recurente din această clasă pot fi considerate nişte automate. Ecuaţia (2.14) calculează starea reţelei la momentele $t = 1, 2, \ldots$, adică descrie secvenţa de *tranziţii ale stărilor*. Reţeaua îşi începe tranziţia la momentul 0 cu $\mathbf{x}^0$ şi evoluează până la o eventuală stare de echilibru care se numeşte *atractor*. Un atractor constă dintr-o singură stare sau dintr-un număr limitat de stări.

2.6 Calcul neural

Aşa cum am văzut în paragrafele anterioare, o reţea neurală calculează ieşirea $\mathbf{y}$ corespunzătoare intrării $\mathbf{x}$ după ce în prealabil au fost stabilite structura reţelei şi valorile ponderilor ei. Acest proces constă de fapt în *regăsirea* informaţiei care a fost înmagazinată anterior.

Să presupunem că o reţea a stocat o mulţime de vectori. Ulterior, dacă reţelei i se prezintă un vector din mulţimea de instruire aceasta îl va asocia cu cel memorat. Un vector incomplet dar asemănător unuia folosit la învăţare va reprezenta un bun punct de pornire pentru regăsire. Acest proces se numeşte *autoasociere*.

O reţea neurală poate să stocheze, însă, asocieri între perechi de vectori astfel încât la regăsire, la prezentarea unui vector sau a unei versiuni uşor distorsionate a lui, reţeaua va genera perechea sa. Acest mecanism se numeşte *heteroasociere*.

Calculul neural poate fi privit şi prin prisma diferenţei dintre *clasificare* şi *recunoaştere*.

Clasificarea este un caz special de heteroasociere. Să presupunem că avem un număr de vectori împărţiţi în mai multe clase. La aplicarea unui vector dintre cei folosiţi la instruire, clasificatorul va genera o informaţie care arată

clasa din care face parte vectorul. Heteroasocierea se face în această situaţie între vector şi codul asociat clasei.

Dacă, însă, vectorul folosit în procesul de regăsire nu se potriveşte exact cu nici unul din mulţimea de instruire, atunci avem de a face cu o recunoaştere. Pattern-ul va fi introdus în clasa pentru care similaritatea este maximă.

Una dintre caracteristicile cele mai importante ale reţelelor neurale este capacitatea de generalizare. O reţea face o bună generalizare atunci când reuşeşte să recunoască în mod corect vectori noi.

2.7 Perceptronul multistrat

Acesta constă dintr-o mulţime de noduri care alcătuiesc stratul de intrare, unul sau mai multe straturi ascunse şi un strat de ieşire, după cum se poate vedea în figura 2.7 [97]. Semnalul de intrare se propagă prin reţea dinspre intrări înspre ieşiri, parcurgând fiecare strat în parte. Perceptronul multistrat aşa cum îl descriem aici reprezintă o generalizare a perceptronului introdus de Rosenblatt.

Perceptronul multistrat a fost folosit cu succes în mai multe aplicaţii cu diverse grade de dificultate prin instruire supervizată folosind algoritmul *back-propagation of error* (*propagarea în urmă a erorii*). Acest algoritm ajustează ponderile unei reţele neurale multistrat pe baza unei metode de minimizare a erorii dintre răspunsul actual al reţelei şi răspunsul dorit.

Un perceptron multistrat are trei caracteristici importante.

a) Modelul fiecărui neuron din reţea include o funcţie neliniară care calculează ieşirea acestuia. Pentru algoritmul de propagare în urmă a erorii este important ca această funcţie să fie diferenţiabilă în orice punct. Câteva exemple de funcţii care satisfac această cerinţă sunt cele din figura 2.6 a), d) şi e).

b) O reţea neurală conţine unul sau mai multe straturi ascunse. Acestea îmbunătăţesc performanţele în învăţare ale reţelei pentru că măresc capacitatea de stocare a caracteristicilor vectorilor de instruire. În acelaşi timp, straturile ascunse sporesc capacitatea de generalizare a reţelelor.

c) Ponderile sinaptice conferă reţelei un grad ridicat de conectivitate, de aceea are un potenţial de procesare paralelă distribuită suficient de mare pentru a rezolva probleme dificile.

Caracteristicile de mai sus împreună cu posibilitatea de a învăţa din experienţă dau forţă acestui model. Primele rezultate obţinute în studierea reţe-

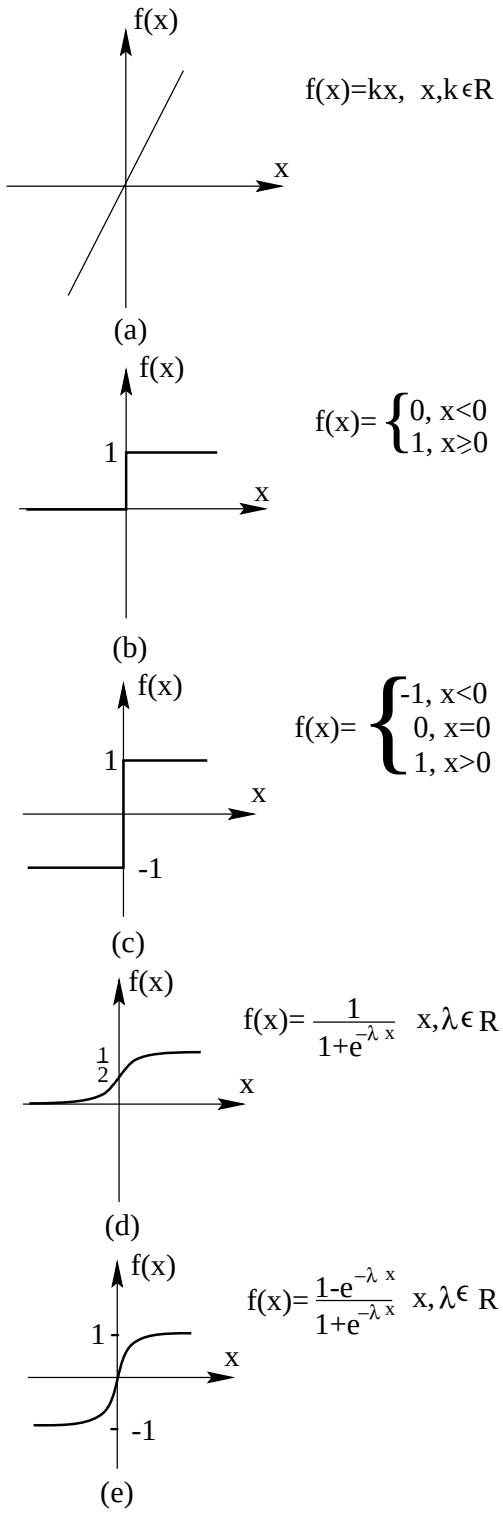

Figura 2.6: Funcţii de transfer: (a)funcţia liniară; (b)funcţia treaptă; (c)funcţia signum; (d)sigmoida unipolară; (e)sigmoida bipolară.

lelor neurale feedforward multistrat au apărut în 1958, atunci când Rosenblatt a prezentat pentru prima dată perceptronul. Evoluţia acestui domeniu s-a dovedit a fi, însă, greoaie datorită lipsei unui algoritm performant de instruire a reţelelor de neuroni artificiali.

Publicarea în anul 1986 a lucrării *Parallel Distributed Processing* [75] de către Rumelhardt şi McClelland a reprezentat o relansare a cercetărilor în domeniul conexionist. Algoritmul *backpropagation* oferă o metodă eficientă de instruire a reţelelor feedforward multistrat, cu toate că nu se poate afirma că ea conduce la o soluţie pentru orice problemă solvabilă.

Pentru a prezenta algoritmul de instruire a perceptronului multistrat prin propagarea în urmă a erorii vom folosi modelul din figura 2.7.

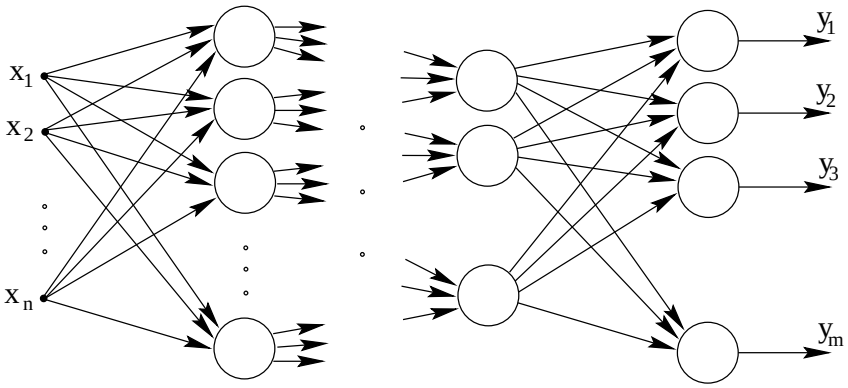

Figura 2.7: Perceptronul multistrat.

În forma generală, reţeaua este complet conectată, ceea ce înseamnă că fiecare neuron din reţea are câte o legătură cu fiecare neuron de pe stratul anterior celui pe care se află el.

În timpul instruirii, reţeaua neurală este parcursă de două tipuri de semnale [30]. *Semnalul funcţiei implementate de reţea* este generat de stimulii aplicaţi la intrare şi se propagă prin fiecare strat în parte, apărând la ieşire sub forma semnalului de ieşire. *Semnalul de eroare* este folosit în algoritmul de instruire şi este calculat pe baza diferenţelor dintre ieşirile actuale ale reţelei şi ieşirile dorite. Nivelul acestui semnal se calculează dinspre ieşiri înspre intrări, de aceea convenim să spunem că semnalul de eroare se propagă în sens invers semnalului funcţiei implementate de reţea.

2.7.1 Algoritmul *backpropagation or error*

Vom prezenta detaliat algoritmul *backpropagation of error* [30] datorită importanţei sale deosebite.

Considerăm că avem o reţea neurală cu un strat de intrare, un strat ascuns şi unul de ieşire, fără a afecta gradul de generalitate a algoritmului. Vom vedea că relaţiile de calcul deduse în acest fel se aplică în mod similar pentru reţele cu oricâte straturi ascunse.

Dorim, aşadar, să aflăm ponderile unei reţele neurale cu un număr fixat de straturi şi de neuroni pe fiecare strat astfel încât aceasta să implementeze funcţia dorită de noi care aplică vectorii de intrare în vectorii de ieşire. Datorită caracterului neliniar al elementelor de procesare pe care le-am numit neuroni, vom deduce o metodă iterativă de calcul a valorii ponderilor. Metoda se bazează pe minimizarea unei funcţii globale de eroare E.

Pentru fiecare neuron calculăm valoarea activării după care, pe baza acesteia, calculăm ieşirea:

$$a_j^p = \sum_{i \in I} w_{ji} y_i^p \tag{2.15}$$

$$y_j^p = f_{act}(a_j^p),\ j \in J. \tag{2.16}$$

În aceste relaţii, p este indicele vectorului curent din mulţimea de instruire pentru care se calculează activarea şi ieşirea. Am notat cu I mulţimea intrărilor neuronului, iar cu J mulţimea ieşirilor tuturor neuronilor din reţea. După cum se vede, ieşirea unui neuron poate deveni intrare pentru alţi neuroni. Pentru neuronii din primul strat ascuns, intrările y_i^p sunt chiar intrările reţelei. Matricea

$$\mathbf{W} = \{w_{ji}\}_{j \in O, i \in I}$$

păstrează valorile ponderilor sinaptice pentru conexiunile dintre neuroni, iar a_j^p reprezintă activarea neuronului j atunci când neuronul primeşte la intrare vectorul p. Mulţimea O este o submulţime a lui J şi conţine ieşirile neuronilor de pe stratul de ieşire. Funcţia de activare f_{act} trebuie să fie derivabilă deoarece algoritmul de propagare în urmă a erorii calculează ponderile sinaptice ţinând cont şi de gradientul lui f_{act}. Cele mai folosite funcţii de activare sunt sigmoida unipolară şi biploară (fig. 2.6). De regulă, fiecare neuron are una dintre intrări fixată la valoarea -1, motiv pentru care ponderea corespunzătoare acesteia poate fi interpretată ca prag al neuronului.

Măsura adaptării reţelei neurale la setul de vectori de instruire este dată de funcţia de eroare. Algoritmul *backpropagation* utilizează funcţia de eroare pătratică medie care are expresia:

$$E^p = \frac{1}{2} \sum_{j \in O} (d_j^p - y_j^p)^2. \tag{2.17}$$

E^p este eroarea individuală pentru un vector, d_j^p este ieşirea dorită pentru neuronul j şi vectorul p, iar y_j^p este ieşirea neuronului j atunci când a fost folosit vectorul p.

Instruirea reţelei constă din modificarea fiecărei ponderi proporţional cu influenţa sa asupra erorii în sensul reducerii lui E:

$$\Delta w_{ji}^p = -\eta \frac{\partial E^p}{\partial w_{ji}}. \tag{2.18}$$

În relaţia de mai sus, η este o constantă numită *rata de instruire*. Semnul minus arată că modificarea ponderii se face în sensul gradientului negativ al erorii.

Relaţia (2.18) poate fi rescrisă ţinând cont de (2.16) astfel:

$$\Delta w_{ji}^p = \eta \frac{\partial E^p}{\partial y_j^p} \frac{\partial y_j^p}{\partial a_j^p} \frac{\partial a_j^p}{\partial w_{ji}}. \tag{2.19}$$

Observăm că:

$$f'_{act}(a_j^p) = \frac{\partial y_j^p}{\partial a_j^p}$$

şi

$$\frac{\partial a_j^p}{\partial w_{ji}} = y_i^p,$$

astfel că relaţia (2.19) devine:

$$\Delta w_{ji}^p = \eta \frac{\partial E^p}{\partial y_j^p} f'_{act}(a_j^p)\, y_i^p. \tag{2.20}$$

În acest moment, se tratează separat cazul în care y_j^p aparţine unui neuron de pe stratul de ieşire şi cel în care aparţine unui neuron din straturile ascunse.

Primul caz, cel în care j este un neuron de ieşire, este mai uşor de rezolvat. Dacă înlocuim (2.17) în (2.20) putem scrie:

$$\Delta w_{ji}^p = \eta\, (d^p - y^p)\, f'_{act}(a_j^p)\, y_i^p, \qquad j \in O. \tag{2.21}$$

A doua situaţie este cea în care neuronul $j \notin O$, adică se găseşte pe un strat ascuns şi nu se poate defini un răspuns dorit pentru el:

$$\Delta w_{ji}^p = \eta \left(\sum_{k \in K} \frac{\partial E^p}{\partial y_k^p} \frac{\partial y_k^p}{\partial a_k^p} \frac{\partial a_k^p}{\partial y_j^p} \right) f'_{act}(a_j^p)\, y_i^p.$$

Relaţia se poate rescrie astfel:

$$\Delta w_{ji}^p = \eta \left(\sum_{k \in K} \frac{\partial E^p}{\partial y_k^p} f'_{act}(a_k^p)\, w_{kj} \right) f'_{act}(a_j^p)\, y_i^p. \tag{2.22}$$

Recursia care apare în această ecuaţie arată că $\frac{\partial E^p}{\partial y_m^p}$ pentru un strat m se calculează în funcţie de valorile $\frac{\partial E^p}{\partial y_n^p}$ obţinute pe stratul n care îl succede pe m. Aceasta înseamnă că ajustările ponderilor se vor calcula din relaţiile de mai sus, parcurgând reţeaua de la ieşiri înspre intrări. În mod obişnuit, se face notaţia:

$$\delta_j^p = \frac{\partial E^p}{\partial a_j^p} = \begin{cases} (d_j^p - y_j^p)\, f'_{act}(a_j^p), & j \in O \\ \sum_{k \in K} (\delta_k^p\, w_{kj})\, f'_{act}(a_j^p), & j \notin O \end{cases},$$

iar δ_j^k se numeşte *gradient local.*

Cu această notaţie, relaţiile (2.21) şi (2.22) se pot scrie într-o formă compactă astfel:

$$\Delta w_{ji}^p = \eta\, \delta_j^p\, y_i^p \tag{2.23}$$

care este regula de calcul ce stă la baza instruirii reţelelor neurale prin propagarea în urmă a erorii.

2.7.2 Parametrii învăţării

Convergenţa algoritmului de instruire prin propagarea în urmă a erorii este asigurată de o serie de factori. Vom prezenta modul în care învăţarea este influenţată de ponderile iniţiale, de constanta de instruire, de factorul momentum şi de modul de ajustare a ponderilor [97].

Ponderile iniţiale

Ponderile reţelei care urmează a fi instruită sunt iniţializate cu valori aleatoare şi mici. Acestea afectează puternic soluţia finală.

Iniţializarea cu valori egale a ponderilor reţelei conduce la o comportare nesatisfăcătoare, cu excepţia cazului când aceastea sunt perturbate în mod aleator în timpul instruirii.

Instruirea unei reţele neurale poate ajunge într-un punct în care eroarea să se stabilizeze la o valoare insuficient de mică sau chiar sa înceapă să crească. Experimentele [30] arată că în astfel de situaţii continuarea instruirii este inutilă. Valorile ponderilor pot începe să crească, în timp ce calitatea reţelei scade. De aceea, se preferă reluarea învăţării prin reiniţializarea ponderilor.

Constanta de instruire

Convergenţa algoritmului *backpropagation* depinde în mod semnificativ de valoarea constantei de instruire η. De obicei, valoarea optimă a acestei constante depinde de problema care trebuie rezolvată.

Deşi instruirea reţelelor neurale prin algoritmul backpropagation este o metodă eficientă, suprafaţa de eroare are, adeseori, proprietăţi care încetinesc convergenţa. În astfel de situaţii, o valoare mai mare a lui η accelerează instruirea. În cazurile în care ajustarea ponderilor se face cu valori prea mari, existând pericolul depăşirii soluţiei şi a unui comportament instabil al reţelei, se preferă micşorarea valorii constantei de învăţare. Aceste consideraţii conduc la concluzia că η trebuie ales experimental pentru fiecare problemă şi că rata de instruire poate fi modificată dinamic în timpul procesului. Cu toate acestea, experimentele [30] au arătat că o valoare cuprinsă între 10^{-3} şi 10 este potrivită pentru majoritatea cazurilor.

Factorul momentum

O metodă de accelerare a convergenţei algoritmului backpropagation şi de prevenire a comportamentului instabil este modificarea regulii delta prin introducerea *factorului momentum* conform formulei:

$$\Delta w_{ji}^{(t)} = \alpha \Delta w_{ji}^{(t-1)} + \eta \delta_j^{(t)} y_i^{(t)}, \tag{2.24}$$

unde t şi $t-1$ indică pasul curent de instruire şi cel anterior, iar α este factorul momentum care este o constantă pozitivă. Primul termen al relaţiei (2.24) se numeşte *termenul momentum* şi scalează cea mai recentă ajustare a ponderilor. După o serie de t paşi de instruire folosind termenul momentum, modificarea actuală se mai poate calcula şi conform formulei:

$$\Delta w_{ji}^{(t)} = -\eta \sum_{k=0}^{t} \alpha^{t-k} \frac{\partial E^{(k)}}{\partial w_{ji}^{(k)}}, \tag{2.25}$$

unde am ţinut cont de (2.18) şi (2.23).

Ajustarea $\Delta w_{ji}^{(t)}$ reprezintă suma unei serii exponenţiale. Pentru ca aceasta să fie convergentă, trebuie impusă condiţia $0 \leqslant |\alpha| < 1$.

Atunci când $\frac{\partial E^{(k)}}{\partial w_{ji}^{(k)}}$ are acelaşi semn de-a lungul mai multor iteraţii consecutive, $\Delta w_{ji}^{(k)}$ creşte iar ponderea w_{ji} este ajustată cu o valoare mai mare. În acest fel, factorul momentum tinde să accelereze convergenţa algoritmului. Atunci când $\frac{\partial E^{(k)}}{\partial w_{ji}^{(k)}}$ are semne opuse în iteraţii consecutive, $\Delta w_{ji}^{(t)}$ scade iar ponderea w_{ji} este ajustată cu o valoare mai mică. În acest caz, factorul momentum are un efect de stabilizare a oscilaţiilor.

Ajustare cumulativă şi ajustare incrementală a ponderilor

Algoritmul de instruire prin propagarea în urmă a erorii poate realiza o mică ajustare a ponderilor după prezentarea fiecărui vector. Această tehnică se numeşte *actualizare incrementală*. Algoritmul backpropagation poate fi conceput, însă, să ajusteze ponderile după o prezentare a întregului set de vectori, caz în care avem de a face cu o *actualizare cumulativă*.

Avantajul folosirii primei metode este acela că algoritmul implementează, într-adevăr, o metodă a gradientului descendent al erorii. În plus, o astfel de modalitate de instruire nu necesită memorarea componentelor calculate pentru fiecare vector care, apoi, să participe la calcularea ajustării globale.

Oricare ar fi modalitatea de instruire folosită, un element foarte important este acela că ordinea în care sunt aleşi vectorii este bine să fie aleatoare.

2.8 Reţele neurale cu întârziere în timp

Reţelele neurale cu întârziere în timp (*time delay neural networks*) [90] sunt o extensie a perceptronului multistrat şi au fost create pentru rezolvarea problemelor de recunoaştere a vorbirii.

Semnalul asociat vorbirii se caracterizează prin modificări rapide în timp, de aceea trebuie prelucrat cu ajutorul unei reţele neurale adaptate acestui comportament. O astfel de reţea neurală trebuie să dispună de o memorie [18] pentru a fi capabilă să surprindă evoluţia în timp a semnalului. Tehnica adoptată pentru a rezolva această problemă constă în introducerea unor elemente de întârziere în structura legăturilor sinaptice ale reţelei.

Arhitectura reţelelor neurale cu întârziere în timp

Neuronul, elementul de bază folosit de o reţea neurală, calculează suma ponderată a intrărilor pe care o transmite apoi unui bloc ce implementează o funcţie neliniară care furnizează răspunsul. Neuronul este modificat într-o reţea neurală cu întârziere în timp prin introducerea celulelor de întârziere $\Delta_1, \Delta_2 \ldots \Delta_N$ (figura 2.8). În felul acesta, anumite componente ale intrării vor influenţa în mod repetat răspunsul neuronului. Se poate spune, astfel, că reţeaua neurală cu întârziere în timp memorează cele mai recente evenimente, fiind adaptată evoluţiei în succesiune temporală a intrărilor, aşa cum este necesar în recunoaşterea vorbirii.

Un vector folosit la instruirea unei reţele neurale pentru recunoaşterea vorbirii are, de regulă, un număr foarte mare de componente. Modelul *time delay neural nerwork* permite ca şi atunci când se folosesc vectori de dimensiuni mari

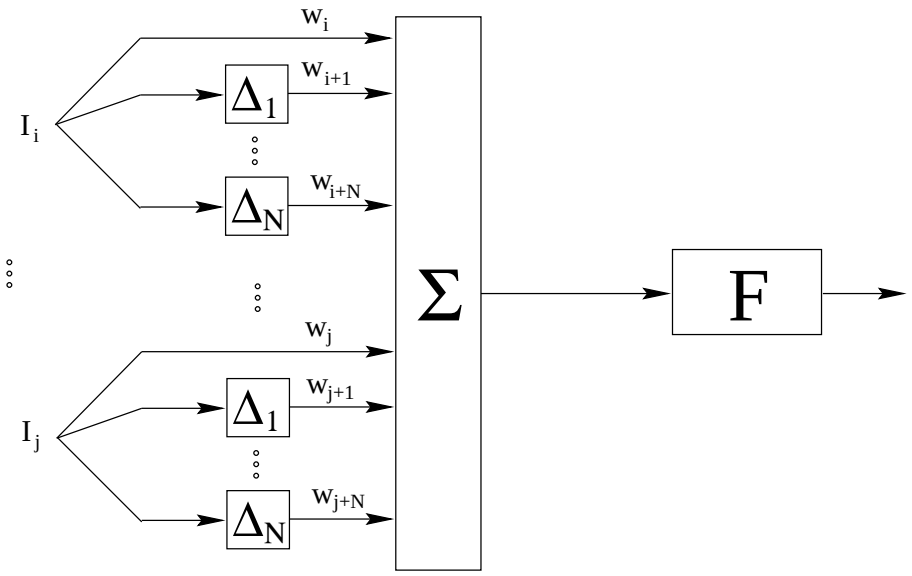

Figura 2.8: Modelul neuronului folosit de reţelele neurale cu întârziere în timp.

numărul de neuroni şi de conexiuni sinaptice să rămână, totuşi, mic. La un moment dat, doar o parte a componentelor vectorului influenţează ieşirea reţelei. Acesta este parcurs de o fereastră care selectează intrarea curentă, aşa cum ilustrează figura 2.9.

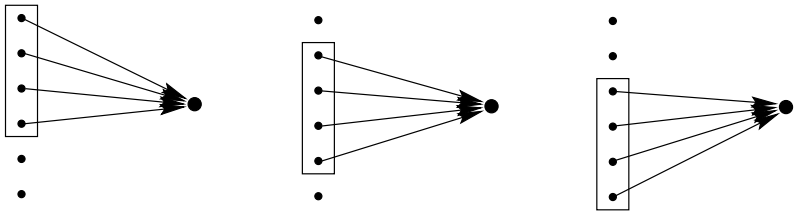

Figura 2.9: Secvenţa în care componentele unui vector devin intrări ale neuronului într-o reţea neurală cu întârziere în timp.

Maniera descrisă mai sus de parcurgere a unui vector poate fi extinsă pentru a construi o reţea neurală cu întârziere în timp ca cea din figura 2.10

Numai componentele vectorului de intrare care se găsesc în fereastră sunt transmise prin intermediul conexiunilor sinaptice spre stratul următor. Fiecărei ferestre dintr-un strat sursă îi corespunde o fereastră în stratul următor. Dimensiunea ferestrei care parcurge un strat este constantă, iar numărul de ferestre din stratul destinaţie este determinat de dimensiunea vectorului care se aplică

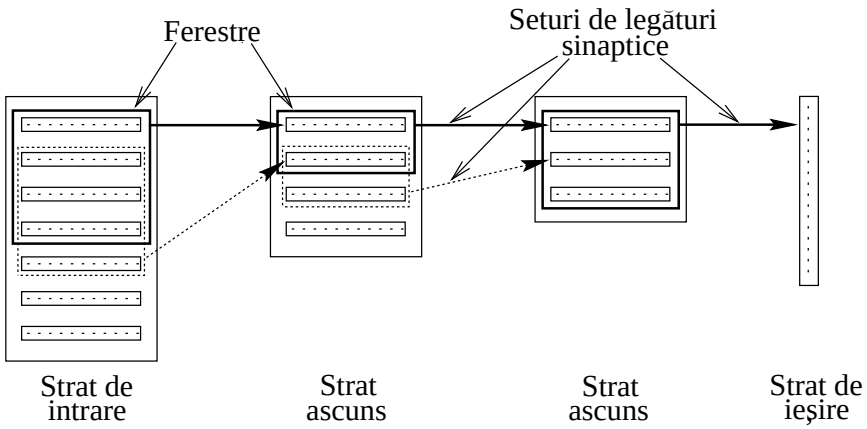

Figura 2.10: Arhitectura unei reţele neurale cu întârziere în timp.

stratului sursă şi de dimensiunea ferestrei care baleiază vectorul.

În exemplul nostru, vectorul este structurat sub forma unei matrici. Fereastra care îl baleiază selectează o secvenţă a lui, aceasta devenind intrarea actuală. Pe baza intrării selectate se calculează ieşirile neuronilor din stratul următor. Fiecare deplasare a ferestrei din stratul de intrare înseamnă adăugarea noilor componente la vectorul care se formează în stratul de ieşire. Fereastra baleiază vectorul sursă până la epuizarea acestuia, deci până când se calculează toate componentele ieşirii din stratul destinaţie.

Instruirea unei reţele neurale cu întârziere în timp

Pentru acest model se foloseşte acelaşi procedeu de învăţare ca şi pentru perceptronul multistrat. Prin instruirea care foloseşte propagarea în urmă a erorii se realizează două treceri prin reţea. Mai întâi, se aplică reţelei neurale un vector de intrare care constituie informaţia propagată înainte. Ponderile sunt ajustate în timpul propagării înapoi a semnalului de eroare care este calculat în funcţie de ieşirile reţelei şi de ieşirile dorite. Această procedură se repetă până când reţeaua converge şi produce ieşirile dorite.

Cele două etape ale algoritmului backpropagation, propagarea înainte şi propagarea înapoi, se desfăşoară ca evenimente separate. Modificarea ponderilor se face prin medierea valorilor calculate pentru fiecare fereastră în parte. Aşadar, un pas de instruire se consideră încheiat doar atunci când ferestrele au baleiat complet intrările tuturor straturilor de neuroni.

Prin această procedură, reţeaua înmagazinează caracteristicile acustico-fonetice ale vectorilor de intrare.

2.9 Reţele neurale cu autoorganizare

2.9.1 Învăţarea hebbiană

O caracteristică importantă a reţelelor neurale este că au capacitatea de a învăţa şi, prin învăţare, îşi îmbunătăţesc performanţele. Atunci când învăţarea este supervizată, este dirijată de perechile intrare-ieşire pe care algoritmul trebuie să le recunoască. Scopul algoritmilor nesupervizaţi sau cu autoorganizare este de a descoperi caracteristicile semnificative ale datelor de intrare fără ca ele să fie transmise din exterior. Aceşti algoritmi înglobează reguli cu caracter local, care influenţează modificările legăturilor sinaptice dintre un neuron şi cei aflaţi în imediata sa vecinătate, permiţând sintetizarea unor corespondenţe între intrări şi ieşiri. Tehnica de adaptare a caracteristicilor reţelei neurale prin autoorganizare se inspiră din neurobiologie într-o măsură mult mai mare decât învăţarea supervizată, deoarece acest proces de organizare este fundamental pentru creier.

Învăţarea hebbiană constă din modificări ale ponderilor sinaptice pe baza interacţiunilor dintre neuronii vecini, o reţea putând avea mai mulţi neuroni activi la un moment dat. Această idee este implementată prin regula Hebb de instruire.

Problema cheie este cum se poate ajunge la o structură viabilă prin autoorganizare, iar răspunsul este dat de Turing [30] care arată că *ordinea globală poate proveni din interacţiuni locale.* Această observaţie ilustrează faptul că interacţiunile locale aparent aleatoare dintre neuronii învecinaţi ai unei reţele pot conduce, în final, la un comportament global coerent, proces care constituie esenţa autoorganizării.

Pentru a ajunge în faza de autoorganizare a reţelei, care este opusă stabilizării, trebuie ca feedback-ul dintre modificările sinaptice şi răspunsul reţelei la stimulii de intrare să fie pozitiv. Primul principiu al autoorganizării formulat de von der Malsburg [30] exprimă acest lucru: *modificările asupra ponderilor sinaptice tind să se autoamplifice.* Autoamplificarea se face pe baza semnalelor locale, adică prin semnalul presinaptic şi cel postsinaptic, acest mecanism fiind, de fapt, o reformulare a postulatului învăţării al lui Hebb.

Pe de altă parte, pentru ca sistemul să se stabilizeze este nevoie de o competiţie pentru un set limitat de resurse care pot fi, de exemplu, resurse energetice sau număr de intrări. Cu alte cuvinte, creşterile unor ponderi sinaptice trebuie compensate de scăderi ale altora care pot chiar să dispară. Această observaţie se poate sintetiza în cel de-al doilea principiu al autoorganizării formulat de von der Malsburg [30]: *limitarea resurselor conduce la competiţie între*

sinapse, rezultând o selecţie a celor mai puternice în detrimentul celor mai puţin adaptate.

Sinapsele nu pot genera singure comportamente favorabile, fiind nevoie de cooperare între sinapsele care converg către un neuron pentru ca acesta să fie activat. Se poate formula, astfel, cel de-al treilea principiu al autoorganizării (von der Malsburg [30]) care arată că *modificările în ponderile sinaptice tind să fie rezultatul cooperării.*

Pentru ca învăţarea prin autoorganizare să fie eficientă, trebuie ca datele de intrare să înglobeze un volum semnificativ de informaţie redundantă. Aceasta este, de fapt, cea care alimentează procesul global de instruire deoarece redundanţa din vectorii de instruire este transferată parametrilor reţelei.

2.9.2 Învăţarea competitivă

Această secţiune prezintă o clasă specială de reţele neurale cunoscute sub numele de *hărţi cu autoorganizare* (*self-organizing maps*)[42]. Ele se bazează pe *învăţarea competitivă* astfel încât, în urma unui proces de selecţie, doar un neuron sau un grup de neuroni vor fi activaţi la un moment dat. Instruirea acestor reţele se bazează pe regula *câştigătorul ia tot.*

Într-o hartă cu autoorganizare, neuronii sunt plasaţi de regulă sub forma unei matrici bidimensionale. În timpul procesului de învăţare ei devin selectivi la anumiţi vectori de intrare sau la anumite clase de vectori. Poziţia neuronilor care răspund fiecărei clase tinde să se stabilizeze pe parcursul instruirii, formându-se o hartă a vectorilor de intrare prin proiecţie bidimensională. Aceşti vectori se vor grupa în urma instruirii conform unor caracteristici interne ale lor, de aceea rezultatul se mai numeşte şi *hartă cu autoorganizare a caracteristicilor.*

Cuantizarea vectorială

Spaţiul intrărilor unei reţele neurale care acceptă ca intrări vectori cu componente reale poate fi considerat infinit. Cuantizarea vectorială este o metodă care produce aproximarea acestui spaţiu printr-un număr K finit de vectori $\mathbf{w}_i,\ i = 1 \dots K$ numiţi şi *prototipuri.* Mulţimea de prototipuri se mai numeşte şi *codebook.* Odată ales acest set de vectori, aproximarea intrării $\mathbf{x}$ înseamnă alegerea vectorului $\mathbf{w}_a$ care este cel mai apropiat de $\mathbf{x}$ în sensul distanţei euclidiene:

$$\|\mathbf{x} - \mathbf{w}_a\| = \min_i \|\mathbf{x} - \mathbf{w}_i\|,\ i = 1 \dots K.$$

Principalul avantaj al cuantizării vectoriale constă în reducerea volumului de informaţie referitoare la mulţimea intrărilor posibile în reţea.

Fiecărui vector $\mathbf{w}_i$, $i = 1 \ldots K$ i se poate asocia o etichetă. În acest fel, alegerea lui $\mathbf{w}_a$ ca cel mai apropiat dintre ei de vectorul de intrare $\mathbf{x}$ înseamnă asignarea etichetei acestui vector lui $\mathbf{x}$, adică recunoaşterea automată a lui $\mathbf{x}$.

Pe de altă parte, pentru că numărul vectorilor $\mathbf{w}_i$, $i = 1 \ldots K$ este finit, alegerea lui $\mathbf{w}_a$ în locul lui $\mathbf{x}$ pentru prelucrările ulterioare implică introducerea unei distorsiuni, adică a unei erori de cuantizare. În mod evident, cu cât K este mai mare, cu atât distorsiunea este mai mică.

Self-organizing feature map

Algoritmul *self-organizing feature map* (SOFM) [42] transformă spaţiul multidimensional al vectorilor de intrare într-unul unidimensional sau bidimensional printr-o grupare conform trăsăturilor comune. În urma instruirii, un vector de intrare va provoca activarea unui grup de neuroni bine definit şi localizat în matricea ieşirilor. Neuronii care se activează sunt organizaţi în jurul unui neuron central care generează o valoare de activare maximă. Cu cât un neuron este mai depărtat de neuronul central, cu atât valoarea lui de activare va fi mai mică.

La construirea unei reţele neurale care implementează o hartă cu autoorganizare trebuie implementat un mecanism care compară rezultatele funcţiilor discriminant ale neuronilor şi îl selectează pe cel cu răspunsul cel mai mare pentru fiecare vector de instruire. Procesul de adaptare trebuie să decurgă în aşa fel încât neuronii selectaţi să devină şi mai bine adaptaţi la vectorii de intrare.

Vom folosi o reţea neurală ca cea din figura 2.11 în scopul de a prezenta algoritmul de instruire a hărţilor cu autoorganizare.

Intrările sunt conectate cu toţi neuronii de pe stratul de ieşire. Notăm vectorii intrărilor

$$\mathbf{x} = [x_1\ x_2 \ldots x_M]^t,$$

iar ieşirile vor fi

$$\mathbf{y} = [y_1\ y_2 \ldots y_N]^t,$$

unde M este numărul intrărilor în reţeaua neurală, iar N numărul elementelor care alcătuiesc harta. Matricea ponderilor va fi:

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \ldots & w_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \ldots & w_{NM} \end{bmatrix}.$$

Pentru găsirea celei mai bune potriviri a vectorului de intrare $\mathbf{x}$ cu matricea ponderilor $\mathbf{w}_j$ corespunzătoare neuronului cu indicele j, se compară produsele

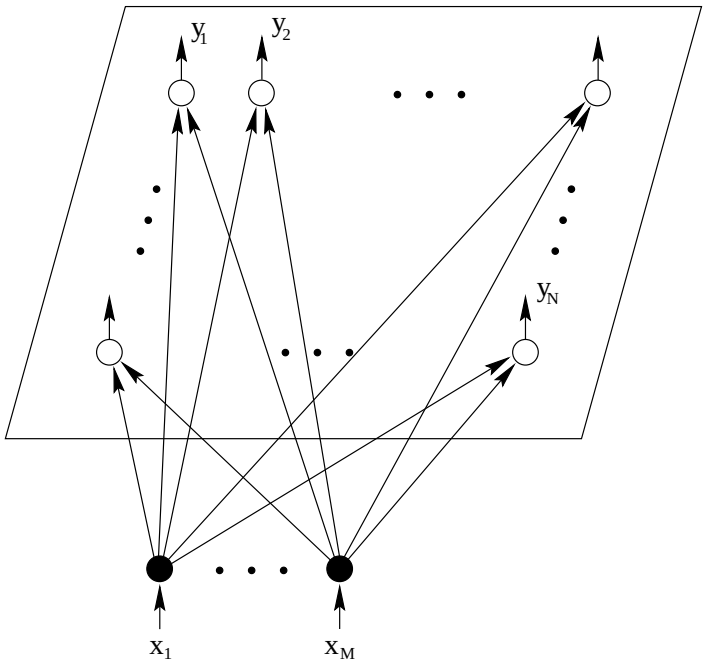

Figura 2.11: Modelul reţelei neurale folosite de SOFM.

scalare $\mathbf{w}_j\mathbf{x}$, $j = 1 \ldots N$ şi se alege cel mai mare. Prin această selecţie se determină centrul zonei cu activare maximă pentru vectorul $\mathbf{x}$.

Algoritmul adaptiv normalizează vectorii de ponderi $\mathbf{w}_j$. În această situaţie, găsirea celei mai bune potriviri poate fi înlocuită cu găsirea distanţei euclidiene minime dintre vectori. Astfel se stabileşte indicele neuronului care îndeplineşte această condiţie, adică:

$$i = \text{ acel } j \text{ pentru care } ||\mathbf{x} - \mathbf{w}_j|| \text{ este minim, } j = 1 \ldots N.$$

Neuronul i care satisface condiţia de mai sus se numeşte *neuron câştigător* pentru patternul $\mathbf{x}$. Prin această relaţie, spaţiul continuu al intrărilor este transformat în spaţiul discret al mulţimii neuronilor. În funcţie de aplicaţia implementată, răspunsul reţelei poate fi ori indicele neuronului câştigător, ori vectorul ponderilor sinaptice corespunzătoare acestuia. În acelaşi context, răspunsul fiecărui neuron poate fi distanţa euclidiană dintre vectorul de intrare şi vectorul său de ponderi.

Algoritmul SOFM foloseşte noţiunea de *vecinătate* a neuronului câştigător (fig. 2.12) care este mulţimea de neuroni din jurul lui asupra cărora se aplică tehnicile de instruire. Dimensiunea vecinătăţii nu este fixată dinainte, iar forma ei poate fi pătrată, circulară etc.

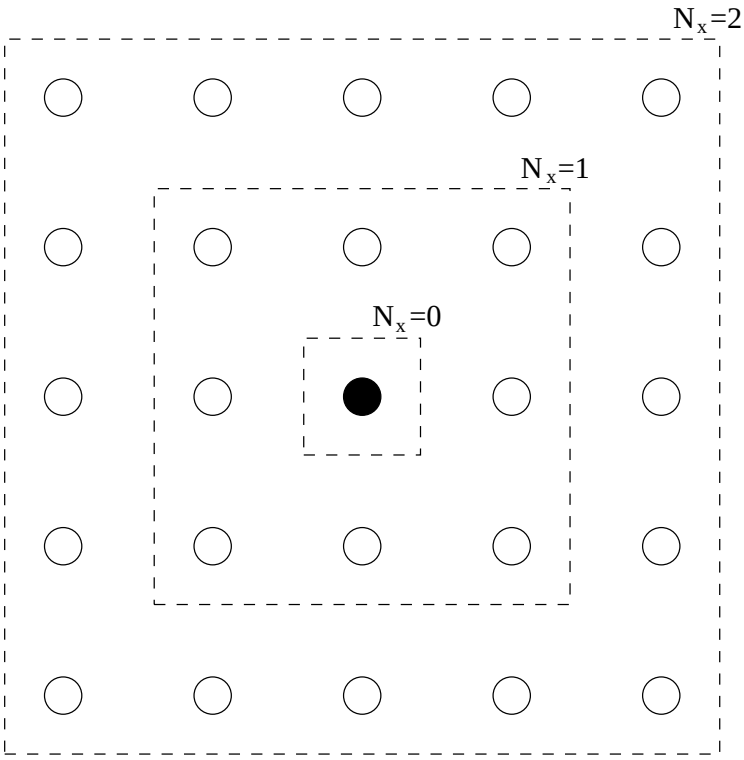

Figura 2.12: Vecinătăţi pătrate ale neuronului câştigător.

Experimental [42] s-a dovedit că alegerea unei vecinătăţi largi care se micşorează în timp conduce la cele mai bune rezultate în instruire. Această modalitate de selecţie a neuronilor asupra cărora se aplică formula de ajustare a ponderilor este echivalentă mai întâi cu o organizare a zonelor de activare pentru fiecare clasă de vectori. Apoi, pe măsura micşorării vecinătăţilor, aceste zone şi, implicit, neuronii câştigători devin din ce în ce mai specializaţi şi mai selectivi. În final, vecinătatea are dimensiune 0, conţinând doar neuronul câştigător.

Pentru ca reţeaua să se autoorganizeze, ponderile sinaptice trebuie să se modifice şi în funcţie de vectorii de intrare. Postulatul lui Hebb al învăţării arată că valoarea unei ponderi sinaptice creşte în funcţie de activităţile presinaptică şi postsinaptică [30]. Această regulă este doar parţial satisfăcătoare. Pentru că modificările au loc doar în sensul creşterii, ponderile finale pot ajunge la saturaţie. Acest lucru este evitat modificând regula lui Hebb prin introducerea unui *termen de uitare* [30]:

$$-g(y_j)\mathbf{w}_j.$$

În expresia de mai sus, $\mathbf{w}_j$ este vectorul ponderilor sinaptice ale neuronului j, iar $g(y_j)$ este o funcţie cu valori pozitive, parametrul său fiind răspunsul

y_j al neuonului. Folosind aceste date, ecuaţia diferenţială care stă la baza formalismului discret este:

$$\frac{\partial \mathbf{w}_j}{\partial t} = \eta y_j \mathbf{x} - g(y_j)\mathbf{w}_j, \quad j = 1, 2 \ldots N. \tag{2.26}$$

În relaţia (2.26), η este rata de instruire. Condiţia care se pune asupra funcţiei g este

$$g(y_j) = 0 \quad \text{pentru } y_j = 0, \forall j = 1, 2 \ldots N,$$

astfel încât pentru un răspuns nul al neuronului să nu se producă nici o modificare a ponderilor.

Experimentele [30] arată că dacă neuronii folosesc o funcţie de activare neliniară care le limitează inferior şi superior ieşirile, după un număr suficient de mare de paşi răspunsul neuronilor din interiorul vecinătăţii va tinde spre limita maximă, iar răspunsul celor din exterior va tinde către cea minimă. În aceste condiţii, se poate considera că:

$$y_j = \begin{cases} 1, & j \text{ activ şi se află în interiorul vecinătăţii} \\ 0, & j \text{ inactiv şi se află în exteriorul vecinătăţii.} \end{cases}$$

Se poate folosi, astfel, o variantă simplificată a funcţiei $g(y_j)$:

$$g(y_j) = \begin{cases} \alpha, & \text{dacă neuronul } j \text{ este activ} \\ 0, & \text{dacă neuronul } j \text{ este inactiv,} \end{cases}$$

unde α este o constantă pozitivă.

Ecuaţia (2.26) devine, în aceste condiţii:

$$\frac{\partial \mathbf{w}_j}{\partial t} = \begin{cases} \eta \mathbf{x} - \alpha \mathbf{w}_j, & j \text{ în interiorul vecinătăţii} \\ 0, & j \text{ în afara vecinătăţii.} \end{cases}$$

Fără a se pierde generalitatea, se alege acelaşi factor de scalare astfel încât $\eta = \alpha$. Relaţia care se obţine arată că vectorul de ponderi $\mathbf{w}_j$ tinde să se apropie de vectorul intrărilor $\mathbf{x}$:

$$\frac{\partial \mathbf{w}_j}{\partial t} = \begin{cases} \eta(\mathbf{x} - \mathbf{w}_j), & j \text{ în interiorul vecinătăţii} \\ 0, & j \text{ în afara vecinătăţii.} \end{cases}$$

Dacă transformăm relaţia continuă într-una discretă, obţinem formula finală care arată cum se calculează ponderile unei hărţi cu autoorganizare:

$$\mathbf{w}_j^{(t+1)} = \begin{cases} \mathbf{w}_j^{(t)} + \eta(\mathbf{x} - \mathbf{w}_j^{(t)}), & j \text{ în interiorul vecinătăţii} \\ \mathbf{w}_j^{(t)}, & j \text{ în afara vecinătăţii,} \end{cases}$$

unde t reprezintă pasul instruirii.

O variantă a algoritmului este cea în care rata de instruire se modifică de-a lungul procesului de instruire. În prima fază, cea de *ordonare*, în care se stabilesc zonele de răspuns pentru fiecare clasă de vectori, rata de instruire este mai mare pentru a permite o evoluţie rapidă. În faza a doua, cea de *convergenţă*, în care ponderile sunt rafinate, rata de instruire este redusă pentru o bună acurateţe a rezultatelor.

Learning vector quantization

Dacă hărţile cu autoorganizare sunt folosite pentru clasificarea vectorilor prin gruparea neuronilor în clase discrete, problema devine una de decizie. Hărţile aproximează semnalele de intrare sau funcţiile de densitate de probabilitate care descriu mulţimile de semnale de intrare prin vectori codebook plasaţi în spaţiul de intrare în aşa fel încât să minimizeze o eroare de cuantizare globală bazată pe distanţa euclidiană. Când semnalele de intrare ajung să fie clasificate, se folosesc mai mulţi vectori codebook pentru a reprezenta fiecare clasă şi identitatea vectorilor în clasă nu mai este atât de importantă. Devine importantă decizia le nivelul frontierei dintre clase. Această strategie introdusă de Kohonen [42] constă dintr-o serie de trei algoritmi intitulaţi generic *learning vector quantization* (LVQ).

Prima variantă numită LVQ1 este cea mai cunoscută şi mai utilizată. Se consideră că fiecărei clase îi sunt asignaţi mai mulţi vectori codebook $\mathbf{w}_j$ şi că fiecare dintre ei este etichetat cu simbolul asociat clasei corespunzătoare. Regiunea corespunzătoare fiecărei clase este definită prin compararea distanţelor euclidiene dintre vectorii de intrare $\mathbf{x}_i$, $i = 1, \ldots, M$ şi prototipurile $\mathbf{w}_j$, $j = 1, \ldots, N$. Eticheta celui mai apropiat prototip $\mathbf{w}_j$ defineşte clasificarea intrării $\mathbf{x}_i$.

Pentru a calcula o poziţionare optimă a prototipurilor $\mathbf{w}_j$, $j = 1, \ldots, N$ în spaţiul intrărilor, se aleg valorile iniţiale ale vectorilor din codebook prin diverse metode, una dintre acestea fiind algoritmul SOFM. Prototipurile sunt apoi etichetate cu ajutorul unui număr de eşantioane pentru care se cunoaşte cărei clase aparţin, eticheta asociată fiecărui prototip fiind dată de eticheta predominantă a intrărilor din vecinătatea sa. Acest procedeu se numeşte *calibrare*. Acurateţea clasificării este îmbunătăţită prin algoritmul iterativ pe care îl prezentăm în continuare. Ideea fundamentală a acestuia este de a îndepărta prototipurile de suprafeţele de decizie în scopul de a delimita clasele cât mai bine.

Fie vectorul de intrare $\mathbf{x}_i$ şi $\mathbf{w}_a$ prototipul cel mai apropiat de el în metrică

euclidiană. Atunci algoritmul LVQ1 de actualizare a lui $\mathbf{w}_a$ este următorul:

$$\begin{aligned}\mathbf{w}_a^{(t+1)} &= \begin{cases} \mathbf{w}_a^{(t)} + \eta(\mathbf{x}_i - \mathbf{w}_j^{(t)}), & \text{când } \mathbf{x}_i \text{ este clasificat corect} \\ \mathbf{w}_a^{(t)} - \eta(\mathbf{x}_i - \mathbf{w}_j^{(t)}), & \text{când } \mathbf{x}_i \text{ nu este clasificat corect} \end{cases} \\ \mathbf{w}_j^{(t+1)} &= \mathbf{w}_j^{(t)}, \quad \text{pentru } j \neq a. \end{aligned} \tag{2.27}$$

Rata de instruire η este un număr real, pozitiv, subunitar ($0 < \eta < 1$) şi poate să fie fixată sau poate să se modifice în timp. În cea de-a doua variantă, se poate porni de la o valoare apropiată de unitate care este micşorată către 0 de-a lungul unui ciclu relativ mare de paşi, de exemplu 100000. Se obţine în felul acesta o metodă de adaptare fină a prototipurilor, care se stabilizează uşor la valorile finale.

Kohonen a introdus o variantă optimizată a acestui algoritm numită *Optimized Learning Vector Quantization* (OLVQ1) [44] care asociază câte o rată de instruire fiecărui prototip, iar rezultatul este o convergenţă mai rapidă:

$$\begin{aligned}\mathbf{w}_a^{(t+1)} &= \begin{cases} \mathbf{w}_a^{(t)} + \eta_a^{(t)}(\mathbf{x}_i - \mathbf{w}_j^{(t)}), & \text{când } \mathbf{x}_i \text{ este clasificat corect} \\ \mathbf{w}_a^{(t)} - \eta_a^{(t)}(\mathbf{x}_i - \mathbf{w}_j^{(t)}), & \text{când } \mathbf{x}_i \text{ nu este clasificat corect} \end{cases} \\ \mathbf{w}_j^{(t+1)} &= \mathbf{w}_j^{(t)}, \quad \text{pentru } j \neq a. \end{aligned} \tag{2.28}$$

Relaţia (2.28) se poate scrie sub forma:

$$\mathbf{w}_a^{(t+1)} = (1 - s^{(t)}\eta_a^{(t)})\mathbf{w}_a^{(t)} + s^{(t)}\eta^{(t)}\mathbf{x}_i^{(t)},$$

unde $s^{(t)} = +1$ dacă $\mathbf{x}_i$ este clasificat corect şi $s^{(t)} = -1$ în caz contrar. Atunci valoarea optimă a ratei de instruire va fi [44]:

$$\eta_a^{(t+1)} = \frac{\eta_a^{(t)}}{1 + s^{(t)}\eta_a^{(t)}}.$$

O variantă modificată a acestui algoritm este LVQ2.1. De această dată, în procesul de instruire se folosesc două prototipuri, $\mathbf{w}_j$ şi $\mathbf{w}_k$, cele mai apropiate de vectorul de intrare $\mathbf{x}_i$, care se găsesc în clase diferite şi care vor fi modificate simultan. Fără a afecta generalitatea problemei, se presupune că cele două clase cărora le aparţin prototipurile au distribuţii diferite, iar planul de decizie care iniţial este stabilit la jumătatea distanţei dintre $\mathbf{w}_j$ şi $\mathbf{w}_k$ trebuie deplasat către poziţia sa reală. Aşa cum se poate vedea şi din figura 2.13, se defineşte o fereastră centrată pe mijlocul acestei distanţe, iar algoritmul se aplică doar vectorilor de intrare care se găsesc în această fereastră.

Dacă se lucrează în spaţii multidimensionale, atunci limitele acestei ferestre sunt hipersfere. Fereastra se defineşte în termenii distanţelor relative d_j şi d_k dintre $\mathbf{x}_i$ şi $\mathbf{w}_j$, respectiv $\mathbf{w}_k$, fiind caracterizată de un parametru constant

$$s = \frac{1 - l}{1 + l},$$

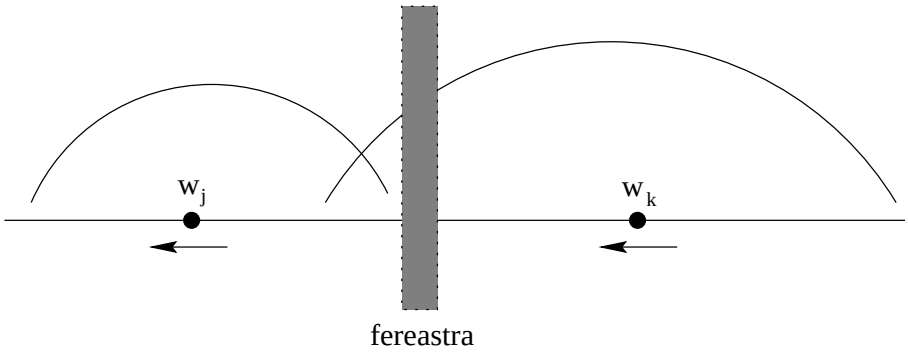

Figura 2.13: Fereastra folosită în algoritmii LVQ2.1 şi LVQ3. Curbele reprezintă distribuţia eşantioanelor din cele două clase.

unde l este lăţimea relativă a ferestrei în punctul său cel mai îngust. Pentru aplicaţiile practice, în [44] se recomandă ca l să ia valori între 0,1 şi 0,2. Se consideră că vectorul $\mathbf{x}_i$ se găseşte în interiorul ferestrei dacă

$$\min\left(\frac{d_j}{d_k}, \frac{d_k}{d_j}\right) > s.$$

Formula de actualizare a prototipurilor conform algoritmului LVQ2.1 este [44]:

$$\begin{aligned} \mathbf{w}_j^{(t+1)} &= \mathbf{w}_j^{(t)} - \eta^{(t)}(\mathbf{x}^{(t)} - \mathbf{w}_j^{(t)}) \\ \mathbf{w}_k^{(t+1)} &= \mathbf{w}_k^{(t)} + \eta^{(t)}(\mathbf{x}^{(t)} - \mathbf{w}_k^{(t)}), \end{aligned} \tag{2.29}$$

unde $\mathbf{w}_j$ şi $\mathbf{w}_k$ sunt prototipurile cele mai apropiate de $\mathbf{x}_i$, în condiţiile în care $\mathbf{x}_i$ şi $\mathbf{w}_k$ aparţin aceleiaşi clase, iar $\mathbf{x}_i$ şi $\mathbf{w}_j$ sunt în clase diferite. În toate celelalte cazuri avem:

$$\mathbf{w}_m^{(t+1)} = \mathbf{w}_m^{(t)},\ m \neq j, k. \tag{2.30}$$

Acest algoritm corectează poziţiile în spaţiu ale prototipurilor $\mathbf{w}_j$ şi $\mathbf{w}_k$ în aşa fel încât planul de decizie poziţionat iniţial la jumătatea distanţei dintre cei doi vectori să se deplaseze către limita dintre cele două distribuţii, făcând ca în final să coincidă cu limita de decizie Bayes.

O altă variantă modificată a algoritmului LVQ este LVQ3. Algoritmul LVQ2.1 se bazează pe ideea de modificare a limitei dintre două clase fără a lua în considerare efectele produse de continuarea rulării acestuia şi după ce poziţia corectă a limitei dintre clase a fost atinsă. În [42] se arată că, deoarece corecţiile se fac proporţional cu distanţele dintre $\mathbf{x}_i$ şi $\mathbf{w}_j$, respectiv $\mathbf{x}_i$ şi $\mathbf{w}_k$, modificarea lui $\mathbf{w}_k$, prototipul care face parte din clasa corectă, este mai amplă decât cea a lui $\mathbf{w}_j$. Rezultatul este o scădere monotonă a distanţei $\|\mathbf{w}_j - \mathbf{w}_k\|$, iar vectorii codebook nu mai sunt plasaţi optim pentru a descrie cu acurateţe

forma graniţelor dintre clase. Din acest motiv, este absolut necesar ca prototipurile să continue să aproximeze distribuţia claselor indiferent de numărul de paşi ai algoritmului. Algoritmul LVQ3 este următorul [44]:

$$\begin{aligned} \mathbf{w}_j^{(t+1)} &= \mathbf{w}_j^{(t)} - \eta^{(t)}(\mathbf{x}^{(t)} - \mathbf{w}_j^{(t)}) \\ \mathbf{w}_k^{(t+1)} &= \mathbf{w}_k^{(t)} + \eta^{(t)}(\mathbf{x}^{(t)} - \mathbf{w}_k^{(t)}), \end{aligned} \tag{2.31}$$

unde $\mathbf{w}_j$ şi $\mathbf{w}_k$ sunt prototipurile cele mai apropiate de $\mathbf{x}_i$, în condiţiile în care $\mathbf{x}_i$ şi $\mathbf{w}_k$ aparţin aceleiaşi clase, iar $\mathbf{x}_i$ şi $\mathbf{w}_j$ sunt în clase diferite. Dacă $\mathbf{x}_i$, $\mathbf{w}_j$ şi $\mathbf{w}_k$ aparţin aceleiaşi clase şi $l \in \{j, k\}$, actualizarea se face conform formulei:

$$\mathbf{w}_l^{(t+1)} = \mathbf{w}_l^{(t)} + \epsilon\eta^{(t)}(\mathbf{x}_i^{(t)} - \mathbf{w}_l^{(t)}). \tag{2.32}$$

Modificarea constă în introducerea unei relaţii de actualizare suplimentare valabilă pentru cazul în care cele două prototipuri aparţin aceleiaşi clase. În [44] se arată că valoarea parametrului ϵ poate fi între 0,1 şi 0,5, iar valoarea optimă depinde de dimensiunea ferestrei, fiind mai mică pentru ferestre mai înguste. Algoritmul are tendinţa de a se stabiliza, iar continuarea instruirii nu afectează plasarea optimă a prototipurilor. Cu toate acestea, în [76] se arată că atunci când se asignează câte un singur prototip fiecărei clase, LVQ3 este similar cu LVQ2.1.

Se impune observaţia că în timp ce LVQ1 actualizează un singur prototip în timpul unui pas de instruire, algoritmii LVQ2.1 şi LVQ3 modifică doi vectori codebook simultan.

2.10 Concluzii

În acest capitol am prezentat modelele neurale fundamentale şi principalele metode care se folosesc pentru instruire. Am văzut care sunt cele mai importante caracteristici ale neuronului McCulloch-Pitts şi ale perceptronului lui Rosenblatt şi am studiat principalele reguli de instruire a unui neuron. Am arătat că reţelele neurale feedforward şi feedback sunt cele mai utilizate arhitecturi. Am prezentat apoi perceptronul multistrat şi algoritmul *backpropagation of error* pentru instruirea supervizată a sa. Reţelele neurale cu întârziere în timp sunt o extensie a perceptronului multistrat şi pot fi folosite pentru instruirea vectorilor cu caracteristici temporale. În final, am văzut că hărţile cu autoorganizare şi algoritmii tip LVQ sunt cele mai importante modele de învăţare competitivă. În capitolul următor vom analiza printr-un studiu comparativ rezultatele obţinute în recunoaşterea cuvintelor izolate cu perceptroni multistrat, cu reţele neurale cu întârziere în timp şi cu hărţi cu autoorganizare.

Capitolul 3

Recunoaşterea vorbirii cu modele clasice de reţele neurale

Acest capitol prezintă un studiu comparativ asupra performanţelor în recunoaşterea cuvintelor izolate şi a vorbitorilor a perceptronilor multistrat, a reţelelor neurale cu întârziere în timp şi a hărţilor cu autoorganizare. Au fost folosite două baze de date care conţin informaţii prelucrate prin analiză cepstrală după înregistrarea semnalelor vocale mai întâi într-un mediu fără perturbaţii, iar apoi după ce au fost transmise printr-o linie telefonică. Cu toate că pentru recunoaşterea cuvintelor izolate rezultatele au fost bune, testele de recunoaştere a vorbitorilor au demonstrat că aceste modele nu oferă întotdeauna soluţii satisfăcătoare.

3.1 Introducere

Problemele computaţionale asociate recunoaşterii vorbirii şi succesul limitat al diverselor tehnici convenţionale au condus la dezvoltarea unor abordări bazate pe reţelele neurale. Metodele descrise în literatură diferă în special prin modul în care sunt convertite semnalele vocale în diverse formate cu scopul de a fi utilizate ca intrări ale reţelelor neurale, prin categoriile de aplicaţii implementate, de exemplu recunoaşterea cuvintelor sau a fonemelor, recunoaştere dependentă sau independentă de vorbitor etc. şi prin tipul de reţea neurală folosită.

Pentru ca o reţea să fie dinamică şi astfel să fie capabilă să proceseze informaţie vocală trebuie să dispună de memorie [18]. Cea mai simplă strate-

gie care poate fi folosită este de a prezenta o secvenţă din fluxul de date la intrarea reţelei neurale. Aceasta este o strategie statică deoarece nu surprinde natura temporală a datelor. Stratul de intrare poate fi privit ca un buffer care păstrează secvenţa temporală curentă. Există două metode prin care se poate schimba conţinutul buffer-ului: informaţia din registru se poate deplasa pentru a fi înlocuită treptat cu date care sosesc pe fluxul de intrare sau poate conţine eşantioane obţinute prin ferestruirea fluxului de date, ferestrele putându-se suprapune. În [7], [23] se foloseşte această abordare care, în mod tipic, are ca suport perceptronul multistrat. Abordarea statică are câteva dezavantaje [18]: impune vectorilor durată fixă, nu ia în considerare caracteristicile temporale ale intrărilor, iar perceptronul multistrat nu se comportă satisfăcător pentru date noi. De altfel, s-a arătat că reţelele neurale feedforward nu sunt capabile să înveţe relaţii temporale care, din acest motiv, trebuie programate în avans [24].

Reţelele neurale *time-delay*, cu întârziere în timp, sunt caracterizate prin faptul că semnalul de intrare este întârziat, iar ieşirile sunt calculate în funcţie de intrările curente şi cele anterioare. În [90] se descrie acest model care este folosit şi în câteva teste de recunoaştere a vorbirii. Modelul prezentat acolo are o proprietate importantă legată de structura dinamică a vorbirii: este invariant la translaţie, adică informaţia învăţată de reţea nu depinde de deplasarea în timp.

Un model diferit de cele anterioare este harta cu autoorganizare [45] care a fost folosită de Kohonen pentru recunoaşterea automată a vorbirii. Cu ajutorul *hărţii fonotopice* (*phonotopic map*) bazate pe acest model, Kohonen a implementat un sistem de transcriere automată a vorbirii numit *phonetic typewriter* capabil să extragă un text după dictare [40].

Majoritarea acestor lucrări oferă rezultate, rate de recunoaştere mai bune decât cele obţinute prin alte tehnici. De exemplu, în [23] este prezentat un sistem de recunoaştere a vorbirii care permite recunoaşterea unui vocabular format din 45 de cuvinte din limba germană, independent de vorbitor. Experimentele descrise aici arată că rata de recunoaştere este de până la 91% pentru vorbitori de acelaşi sex şi de până la 72% pentru un set de date care cuprinde cuvinte pronunţate atât de bărbaţi cât şi de femei. Este, însă, dificil să comparăm în mod direct rezultatele obţinute de diverşi autori pentru că în majoritatea cazurilor acestea se bazează pe seturi particulare de date. Este de dorit ca pentru toate modelele testate să folosim aceleaşi date de instruire, obţinând astfel o imagine mult mai fidelă a performanţelor lor. Scopul acestui capitol este de a prezenta o comparaţie a rezultatelor obţinute în recunoaşterea vorbirii prin trei tehnici diferite [66]: perceptronul multistrat (MLP), reţeaua

neurală cu întârziere în timp (TDNN) şi harta cu autoorganizare (SOM), pentru aceeaşi bază de date. Pentru că aceste tipuri de reţele neurale sunt foarte cunoscute, le-am asociat atributul "clasice".

3.2 Bazele de date Speechdata şi Telephdata

În testele de recunoaştere a vorbirii cu MLP, TDNN şi SOM am folosit două baze de date care au fost realizate la Northern Ireland BioEngineering Center (NIBEC). Ele cuprind pronunţiile în limba engleză ale cifrelor 0 până la 9 la care au fost adăugate *nought*, *oh* şi *zero* care sunt variante ale cifrei 0. Pentru crearea fiecărei baze de date au fost alese câte 25 de persoane care au repetat cele 12 cuvinte de câte 20 de ori, rezultând un număr total de 6000 de pronunţii. La alcătuirea primei baze de date au fost aleşi 19 bărbaţi şi 6 femei, iar pentru cea de-a doua 15 bărbaţi şi 10 femei. Toate persoanele au avut vârsta cuprinsă între 20 şi 60 de ani şi au folosit limba engleză cu accent nord-irlandez.

Pentru prima bază de date care se numeşte Speechdata, informaţiile au fost înregistrate într-un mediu controlat, în care zgomotele şi interferenţele au fost reduse la minim. Fiecare vorbitor a fost înregistrat într-un studio izolat fonic, pe un casetofon profesional, cu un microfon de înaltă calitate. Semnalul vocal a fost apoi digitizat folosind un convertor analog/digital pe 12 biţi cu frecvenţa de eşantionare de 7,5 KHz, după care informaţia a fost stocată pe suport magnetic.

Cea de-a doua bază de date numită Telephdata este destinată recunoaşterii automate a vorbirii printr-o reţea telefonică, fiind mai apropiată de condiţiile unei aplicaţii reale. Semnalul vocal a fost, de această dată, captat şi stocat în timp real. Vorbitorii au fost aşezaţi pe rând la un pupitru dotat cu un aparat telefonic şi un calculator. Ei au pronunţat la telefon cuvintele pe care le-au citit de pe ecran. Semnalul telefonic recepţionat a fost introdus în sistemul de digitizare prezentat mai sus.

Crearea mulţimii de vectori se bazează pe folosirea unui banc de filtre ale cărui benzi de trecere sunt calculate conform unei scale neliniare numită *scala mel*. Ieşirile generate de acest banc de filtre au fost proiectate în domeniul cepstral printr-o transformare cosinus discretă. Coeficienţii cepstrali împreună cu o colecţie de *coeficienţi delta*, calculaţi pe baza unor diferenţe ale primilor, alcătuiesc informaţia folosită la instruirea reţelelor neurale.

Oricare ar fi sistemul de recunoaştere a vorbirii folosit, etapa de preprocesare a semnalului vocal este foarte importantă. Scopul ei este de a extrage componentele de bază ale semnalului acustic şi de a le coda într-un set compact şi eficient de parametri.

Scala de frecvenţe *mel* a fost realizată în urma studierii modului în care

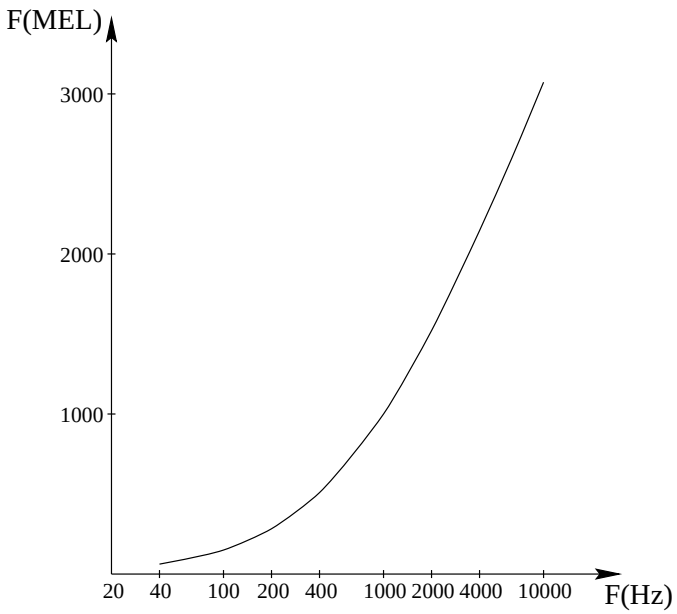

Figura 3.1: Scala de frecvenţe *mel* realizată în urma studierii modului de percepţie a sunetelor de sistemul auditiv uman.

sunetele sunt percepute de sistemul auditiv uman. *Mel* este o unitate de măsură a înălţimii percepute de ureche umană pentru un ton. Este cunoscut faptul că sunetele de frecvenţe scăzute sunt distinse mai greu de către om. La fel se întâmplă şi cu sunetele de frecvenţă înaltă. În funcţie de nivelul de antrenare şi de starea de sănătate, frecvenţele maxime audibile pot să difere de la un individ la altul. Stevens şi Volkman au stabilit în 1940 această scală pe baza unui experiment descris în continuare [16]. În mod arbitrar au stabilit frecvenţa de referinţă de 1000 Hz şi i-au atribuit 1000 mel. Ascultătorilor li s-a cerut apoi să modifice frecvenţa fizică a sunetului până când intensitatea percepută a devenit de două ori mai mare decât referinţa. Această frecvenţă a fost etichetată cu 2000 mel. Repetând procedeul pentru frecvenţe mai mari de 1000 Hz, dar şi mai mici decât acest prag, a fost stabilită prin mediere scala din figura 3.1.

Aceasta este aproximativ liniară pentru frecvenţe sub 1 KHz şi logaritmică peste acest nivel. Pentru aproximarea ei, în faza de preprocesare a bazei de date s-a folosit relaţia:

$$F_{\mathrm{MEL}} = 2595 \log \left(1 + \frac{F_{\mathrm{Hz}}}{700}\right).$$

Pornind de la rezultatul prezentat mai sus, a fost realizat un banc de filtre, construcţie care se bazează pe capacitatea sistemului auditiv de a percepe diferitele frecvenţe din spectrul audio. Pentru frecvenţe sub 1 KHz lăţimile

benzilor sunt de 100 Hz şi cresc logaritmic peste 1KHz (figura 3.2).

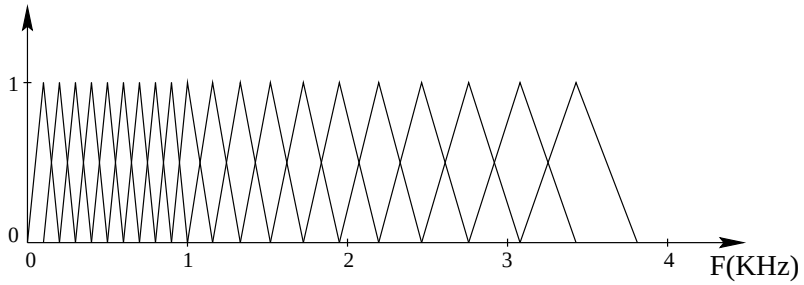

Figura 3.2: Banc de filtre a cărui construcţie se bazează pe capacitatea sistemului auditiv uman de a percepe diferitele frecvenţe din spectrul audio.

Semnalul vocal a fost împărţit în cadre de 20 ms folosind ferestre Hamming. Fiind suprapuse, durata lor a fost stabilită la 30 ms. Semnalul ferestruit a fost aplicat bancului de filtre, iar ieşirea fiecăruia dintre ele, X_j, a fost obţinută prin agregarea componentelor spectrale din banda de frecvenţă j.

Conform modelului general acceptat, semnalul vocal este alcătuit dintr-o secvenţă de excitaţie compusă printr-o relaţie de convoluţie cu răspunsul la impuls al tractului vocal. Noi avem acces doar la semnalul de ieşire, însă este deosebit de utilă eliminarea uneia dintre componente pentru studiul celeilalte. Deoarece combinaţia rezultată nu este liniară, separarea devine o problemă destul de dificilă.

Să presupunem că dorim să studiem un semnal $x(n)$ de frecvenţă joasă care este perturbat prin suprapunere de un semnal de frecvenţă înaltă $w(n)$. Semnalul rezultat va fi $y(n) = x(n) + w(n)$. Putem folosi transformata Fourier care este un operator liniar pentru a transfera semnalul din domeniul timp în domeniul frecvenţă. Spectrul este o reprezentare a semnalului în care putem separa acele componente specifice semnalului util de cele ale perturbaţiei. Dacă scopul este studiul semnalului $x(n)$, putem folosi un filtru trece-jos care extrage componentele dorite. Considerând că O este un operator care realizează filtrarea, atunci

$$O\{y(n)\} = O\{x(n) + w(n)\} = O\{x(n)\} + O\{w(n)\} \approx x(n).$$

Atunci când semnalele sunt combinate prin convoluţie, nu se poate trage o concluzie despre influenţa filtrului asupra fiecărei componente. De aceea a fost adoptată utilizarea cepstrului în probleme de recunoaşterea vorbirii [16], [72]. Folosind această metodă, componentele semnalului vor fi separate în cepstru şi

vor fi combinate liniar. Pentru un semnal $y(n)$, cepstrul se defineşte astfel:

$$c_y(n) = F^{-1}\{\log|F\{y(n)\}|\} = \frac{1}{2\pi}\int_{-\pi}^{\pi}\log|Y(\omega)|e^{j\omega n}\mathrm{d}\omega$$

unde $F\{\cdot\}$ este o transformare Fourier discretă. Pentru că $\log|Y(\omega)|$ este par, $F\{\cdot\}$ se înlocuieşte cu o transformare cosinus.

Pe baza acestor consideraţii, coeficienţii cepstrali folosiţi în bazele de date testate au fost calculaţi conform următoarei relaţii:

$$C_i = \sum_{j=1}^{N}\left(\log|X_j|\cdot\cos\left(\frac{\pi i}{N}(j-0,5)\right)\right), \quad 1 \leqslant i \leqslant M$$

unde N este numărul de filtre din bancul de filtre, iar M este numărul dorit de coeficienţi cepstrali. Fiecare coeficient delta, d_P, al cadrului P se calculează conform formulei [16]:

$$d_P = \frac{\sum_{i=1}^{M} i(C_{P+i} - C_{P-i})}{2\sum_{i=1}^{M} i^2},$$

unde C_P este coeficientul cepstral pentru cadrul P. Pentru începutul şi sfârşitul cuvântului se folosesc relaţiile [16]:

$$d_P = C_{P+1} - C_P, \quad P < M$$

şi

$$d_P = C_P - C_{P-1}, \quad P \geqslant N_F - M$$

unde N_F este numărul de cadre din cuvânt.

Pentru Speechdata şi Telephdata, valorile lui N şi M au fost 16 şi 8, vectorul unui cadru având 18 componente care constau din 8 coeficienţi cepstrali, 1 coeficient numit *log-energy* obţinut prin logaritmarea energiei semnalului din fereastra Hamming curentă şi 9 coeficienţi delta. Semnalul vocal pentru fiecare cuvânt a fost procesat în 15 ferestre Hamming şi s-au obţinut câte 270 de coeficienţi pentru un cuvânt.

3.3 Experimente

Vom descrie în acest subcapitol rezultatele obţinute la testarea capacităţii de recunoaştere a perceptronului multistrat, a reţelelor neurale cu întârziere în timp şi a hărţilor cu autoorganizare folosind bazele de date Speechdata şi Telephdata.

De regulă, condiţiile experimentelor nu sunt aceleaşi şi compararea obiectivă a performanţelor diverselor tipuri de reţele neurale este dificil de realizat prin

sintetizarea rezultatelor disponibile în literatură. Atunci când bazele de date diferă, comparaţia nu se poate face corect. Acesta este motivul pentru care am considerat necesară o analiză pe baza unui singur set de date.

Experimentele prezentate în acest paragraf au urmărit două obiective: testarea performanţelor acestor reţele neurale în recunoaşterea vorbirii şi în recunoaşterea vorbitorilor. Am separat datele de instruire disponibile în cele două baze de date într-un set folosit la instruire şi altul folosit la recunoaştere. În setul de antrenare am inclus 40% din date, iar la testare le-am utilizat pe celelalte 60%.

Reţelele neurale pe care le-am testat în acest capitol sunt modele de referinţă în domeniu şi a fost dezvoltată o paletă foarte largă de implementări ale algoritmilor lor de instruire. Multe dintre aceste implementări sunt publice şi sunt disponibile in reţeaua Internet cu scopul declarat de a facilita activitatea de cercetare.

Stuttgart Neural Network Simulator (SNNS) [98] este un mediu eficient şi flexibil pe care l-am folosit la implementarea perceptronilor multistrat şi a reţelelor neurale cu întârziere în timp. Are patru componente importante: nucleul de simulare, interfaţa grafică (*GUI*), simulatorul programabil (*Batchman*) şi compilatorul (*snns2c*). Nucleul de simulare realizează operaţiile asupra reţelelor neurale. Interfaţa grafică este un modul care controlează simularea în timpul rulării. Oferă, de asemenea, o reprezentare grafică a reţelelor neurale. Interfaţa grafică poate fi folosită direct pentru a crea, manipula şi vizualiza reţelele neurale în diverse moduri. Simulatorul foloseşte un limbaj de programare special, integrat în SNNS. Permite execuţia operaţiilor în background şi implementarea reţelelor neurale fără interfaţa grafică. Acest limbaj de programare este similar unui limbaj de nivel înalt. Toate opţiunile care sunt accesibile prin interfaţa grafică sunt disponibile şi în *Batchman*.

Pachetul de programe SOM_PAK [43] a fost realizat de o echipă condusă de T. Kohonen şi implementează toate operaţiile necesare unei rulări corecte a instruirii şi aplicării hărţilor cu autoorganizare. Am folosit acest set de programe în experimentele de recunoaştere a cuvintelor izolate si a vorbitorilor cu SOM. Modulele de iniţializare (*randinit*, *lininit*), instruire (*vsom*) şi testare (*qerror*) pot fi rulate atât automat, folosind programul *vfind* care solicită în mod interactiv toţi parametrii necesari, sau separat, direct de la consolă. Acest pachet oferă, în plus, o serie de module pentru calibrare, vizualizare şi testare a hărţilor cu autoorganizare.

3.3.1 Recunoaşterea cuvintelor

Cele două baze de date conţin câte 12 cuvinte pronunţate de câte 25 de vorbitori. Fiecare vorbitor a repetat fiecare cuvânt de câte 20 de ori, astfel încât atât Speechdata cât şi Telephdata conţin câte 6000 de repetiţii. De fiecare dată primele 8 repetiţii ale fiecărui cuvânt au fost folosite la instruire şi ultimele 12 la test. Datorită metodei de creare a bazelor de date, vectorii au fost organizaţi sub forma unor matrici de 15x18 componente, deci reţelele neurale folosite au avut câte 270 de intrări.

Atât pentru perceptronul multistrat cât şi pentru reţelele neurale cu întârziere în timp am folosit configuraţii cu un singur strat ascuns. Numărul de neuroni de pe stratul ascuns a fost de aproximativ 100 de neuroni, cu mici variaţii care nu au influenţat semnificativ capacitatea de învăţare.

Pentru că numărul de cuvinte este 12, am stabilit ca acesta să fie şi numărul de neuroni de ieşire pentru MLP şi TDNN. Fiecare ieşire se activează pentru câte o singură clasă de vectori, generând o ieşire apropiată de 1. Ieşirile care nu se activează generează valori apropiate de 0. În faza de instruire am folosit chiar valorile 1 şi 0 pentru ieşirile dorite, iar la testare am aceptat un prag de 0,3. Aceasta înseamnă că o ieşire între 0,7 şi 1 este asimilată lui 1, iar una între 0 şi 0,3 este 0.

Hărţile cu autoorganizare folosite pentru recunoaşterea cuvintelor au avut 17x15 noduri. Instruirea a avut loc în două etape. Pentru prima, cea de organizare a hărţii, am rulat 10000 de paşi ai algoritmului, iar pentru a doua, cea a accentuării zonelor, am parcurs 120000 de paşi. Prin calibrare se stabileşte pe baza distanţei euclidiene clasa căreia îi aparţine fiecare neuron. Aceasta se realizează statistic, prin prezentarea la intrările hărţii instruite a unor vectori despre care se ştie din ce clase fac parte. Nodurile rămase neetichetate nu sunt folosite în faza de test.

Rezultatele experimentelor de recunoaştere a cuvintelor pentru fiecare tip de reţea neurală şi pentru ambele baze de date sunt prezentate în tabelul 3.1.

Tabelul 3.1: Rezultatele experimentelor de recunoaştere a cuvintelor.

	MLP	TDNN	SOM
Speechdata	93,3%	95,1%	97,1%
Telephdata	76,3%	90,5%	96,8%

3.3.2 Recunoaşterea vorbitorilor

O problemă mai dificilă pe care am implementat-o cu modelele clasice de reţele neurale a fost recunoaşterea vorbitorilor. Ne-am propus să instruim MLP şi TDNN cu vectori grupaţi după criteriul identităţii persoanei. Astfel, cuvintele din cele două baze de date au fost separate în 25 de clase care corespund celor 25 de vorbitori. Acum nu ne mai interesează care sunt cuvintele pe care le-au pronunţat subiecţii pentru că această informaţie devine nerelevantă. Repetarea, însă, a aceloraşi cuvinte produce redundanţa necesară algoritmilor de instruire care pot extrage trăsăturile dominante ale vocii fiecărui vorbitor. Răspunsurile dorite ale vectorilor de instruire au fost ca şi la recunoaşterea cuvintelor secvenţe binare cu o componentă care are valoarea 1 şi care arată clasa din care face parte vectorul, iar restul au valoarea 0. Fiind vorba de 25 de clase, stratul de ieşire al reţelelor neurale a fost redimensionat la această valoare. Stratul ascuns a crescut la 150 de noduri deoarece am constatat experimental că este soluţia optimă.

În cazul instruirii SOM am optat pentru o reţea de 30x30 de noduri care a fost calibrată cu vectori din cele 25 de clase.

Sinteza rezultatelor obţinute pentru acest set de experimente se găseşte în tabelul 3.2.

Tabelul 3.2: Rezultatele experimentelor de recunoaştere a vorbitorilor.

	MLP	TDNN	SOM
Speechdata	71,2%	57,2%	52,8%
Telephdata	6,7%	32,33%	44,7%

3.4 Concluzii

Rezultatele experimentale prezentate în acest capitol creează o imagine a capacităţii unor modele clasice de reţele neurale de a rezolva probleme de recunoaştere a vorbirii.

Recunoaşterea cuvintelor izolate din Speechdata a condus la rezultate satisfăcătoare. La recunoaşterea cuvintelor din Telephdata, perceptronul multistrat a avut o comportare foarte slabă. Modul de construire a bazei de date a permis MLP obţinerea unui rezultat apropiat de TDNN, chiar dacă structura reţelei neurale nu permite extragerea unor informaţii legate de secvenţa în timp a cadrelor ce alcătuiesc un cuvânt. Performanţele SOM arată potenţialul crescut

al reţelelor neurale cu autoorganizare. Studiind ratele de recunoaştere, putem trage concluzia că modelul standard MLP nu este potrivit pentru recunoaşterea eficientă a cuvintelor izolate.

Pe de altă parte, rezultatele obţinute la testele de recunoaşterea vorbitorilor arată că modelele clasice sunt insuficiente. Ratele de succes de până la 50% dovedesc faptul că ele nu sunt capabile să extragă caracteristicile vocale ale indivizilor într-o măsură suficient de bună şi este nevoie de implementarea unor tehnici care să răspundă mai bine cerinţelor. Pentru a îmbunătăţi performanţele reţelelor neurale în recunoaşterea vorbitorilor, în capitolul următor vom introduce reţelele neurale concurente, un nou model de recunoaştere care foloseşte ideea de modularitate şi principiul *divide et impera*.

Capitolul 4

Reţele neurale concurente

Propunem în acest capitol un nou model neural de recunoaştere numit Concurrent Neural Networks (CNN) care constă dintr-o colecţie de reţele neurale, iar decizia la clasificare se ia prin regula câştigătorul ia tot. Fiecare reţea neurală componentă a sistemului este instruită separat pentru a răspunde corect intrărilor dintr-o singură clasă. Am aplicat acest model la recunoaşterea vorbitorilor şi a vocalelor folosind perceproni multistrat, reţele neurale cu întârziere în timp şi hărţi cu autoorganizare ca module ale sistemului concurent.

4.1 Introducere

Experimentele de recunoaştere a vorbitorilor descrise în capitolul anterior au demonstrat că modelele clasice pe care le-am testat nu oferă întotdeauna rezultate mulţumitoare. Perceptronul multistrat conduce uneori la rezultate mai slabe datorită faptului că arhitectura sa are un caracter general şi nu exploatează caracteristicile semnalului vocal.

Performanţele acestor reţele scad foarte mult pentru recunoaşterea vorbitorilor. În experimentele noastre, numărul claselor folosite la recunoaşterea vorbitorilor este dublu faţă de cel folosit pentru recunoaşterea cuvintelor. Reducând numărul de vorbitori, se constată, însă, că performanţele rămân inferioare celor stabilite la recunoaşterea cuvintelor, ceea ce arată că problema este una mai complexă.

Pe de altă parte, este evident că reducerea numărului de vorbitori provoacă îmbunătăţirea comportamentului reţelelor neurale prin creşterea ratei de succes pentru că problema pe care o implementează devine mai puţin dificilă. Problema globală poate fi redusă la o sumă de subprobleme prin separarea vorbitorilor în grupuri mai mici în interiorul cărora recunoaşterea este mai uşor de realizat.

Reţelele neurale modulare [30] au la bază ideea că, pe lângă nivelele ierarhice de organizare a reţelelor neurale artificiale: sinapse, neuroni, straturi de neuroni şi reţeaua însăşi, se poate crea un nou nivel care combină mai multe reţele neurale. Modularitatea are o justificare neurobiologică, fiind un principiu important în arhitectura sistemelor nervoase ale vertebratelor. De altfel, sistemul vizual oferă un bun exemplu de calcul modular deoarece imaginile percepute sunt descompuse în imagini de dimensiuni reduse care sunt analizate separat, iar rezultatele individuale sunt apoi combinate.

Haykin [30] arată că o reţea neurală se numeşte modulară atunci când calculul realizat de reţea poate fi descompus în două sau mai multe module care operează pe seturi distincte de intrări, fără a comunica între ele. Ieşirile modulelor sunt mediate de o unitate de integrare care nu permite transmiterea informaţiei înapoi către module. Unitatea de integrare trebuie să decidă cum se combină ieşirile modulelor pentru a forma ieşirea finală a sistemului şi trebuie să decidă ce seturi de vectori trebuie învăţate de fiecare modul. Modularitatea poate fi privită ca o formă de implementare a principiului *divide et impera* care permite rezolvarea problemelor complexe prin descompunerea acestora în subprobleme şi combinarea soluţiilor individuale.

Reţele neurale modulare au fost studiate în diverse contexte. Rueckl, Cave şi Kosslyn pun în [74] problema identificării unor obiecte cunoscute şi a găsirii locaţiei acestora în spaţiu. S-a constatat că timpul de instruire a fost mai scurt atunci când s-au folosit două reţele neurale în locul uneia singure. Tehnica numită *connectionist glue* propusă de Waibel, Sawai şi Shikano [91] presupune instruirea individuală a reţelelor neurale pentru diferite probleme de recunoaştere a vorbirii urmată de regruparea lor. Jacobs, Jordan şi Barto [34] instruiesc un sistem modular în care modulele concurează pentru a învăţa diverşi vectori particulari. Anand, Mehora, Mohan şi Ranka [1] propun o metodă prin care problema clasificării vectorilor în K clase este transformată în K probleme de clasificare a vectorilor în două clase. Modulele decid dacă un vector aparţine unei clase sau nu. Algoritmul backpropagation este modificat pentru a suplini diferenţa mare dintre exemplele pozitive şi cele negative pe care o presupune această soluţie.

Modelul propus de noi [11] numit *Concurrent Neural Networks (CNN)* introduce o nouă tehnică neurală de recunoaştere care se bazează pe ideea competiţiei între mai multe reţele neurale care lucrează în paralel, decizia finală luându-se pe baza regulii *câştigătorul ia tot*. Numărul de reţele folosit este egal cu numărul de clase în care sunt grupaţi vectorii, iar instruirea este supervizată. Fiecare reţea este proiectată să recunoască corect vectori dintr-o singură clasă, astfel că răspunsurile cele mai bune apar doar atunci când îi sunt prezentaţi vectori

din clasa cu care a fost instruită. Acest model este de fapt un cadru care oferă flexibilitate arhitecturii fiindcă modulele pot fi reprezentate de diverse tipuri de reţele neurale.

Pornind de la modelul CNN din [11], Neagoe [60] a dezvoltat şi experimentat modelul *Concurrent Self-Organizing Maps (CSOM)* care se detaşează ca o tehnică nouă cu performanţe excelente. A fost demonstrată importanţa CSOM pentru cazul general al vectorilor cu repartiţii normale multimodale şi s-a arătat că poate fi folosit cu succes pentru clase largi de probleme de recunoaştere a formelor. CSOM a fost aplicat cu succes de Neagoe [60] pentru recunoaşterea feţelor umane şi de Neagoe şi Ropot în clasificarea imaginilor satelitare [61], în aplicaţii de clasificare a formelor [62], pentru recunoaşterea feţelor umane şi la recunoaşterea vorbitorilor [64].

Ideea competiţiei dintre hărţile cu autoorganizare a fost utilizată ulterior şi de Lapidot, Guterman şi Cohen în [51] pentru recunoaşterea nesupervizată a vorbitorilor. Fiecare vorbitor este modelat printr-o hartă, în timp ce un algoritm iterativ permite ajustarea hărţilor.

Acest capitol prezintă noul model al reţelelor neurale concurente şi analizează performanţele sale pentru recunoaşterea vorbitorilor şi a vocalelor.

4.2 Reţele neurale concurente

Schema generală folosită pentru instruirea reţelelor neurale concurente este cea din figura 4.1.

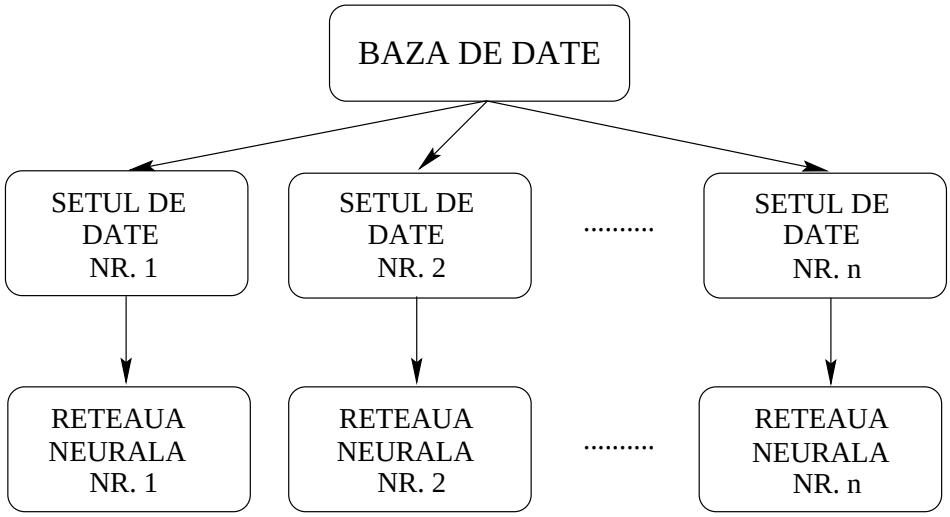

Figura 4.1: Modelul folosit la instruirea reţelelor neurale concurente.

În această schemă, n reprezintă numărul de reţele neurale care lucrează în paralel, dar este egal totodată şi cu numărul de clase în care sunt grupaţi vectorii de instruire.

Considerăm mulţimea $X \in R^p$ formată din M vectori p-dimensionali, adică:

$$X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\},\ \mathbf{x}_i \in R^p,\ 1 \leqslant i \leqslant M.$$

Presupunem că pentru fiecare vector din mulţimea X se cunoaşte apartenenţa la una dintre cele n clase, unde:

$$X = X_1 \cup X_2 \cup \ldots \cup X_n$$

şi

$$X_1 \cap X_2 \cap \ldots \cap X_n = \Phi,$$

iar $X_1, X_2, \ldots, X_n$ sunt submulţimi ale mulţimii X şi conţin vectorii de intrare pentru fiecare dintre cele n reţele neurale.

Mulţimea X de vectori este obţinută în urma preprocesării semnalelor vocale achiziţionate în scopul instruirii reţelelor. Din această mulţime sunt extrase seturile de vectori $X_j,\ j = 1, 2 \ldots n$ cu care vor fi instruite cele n reţele neurale. În urma aplicării procedurii de învăţare, fiecare reţea neurală va trebui să răspundă pozitiv unei singure clase de vectori şi să dea răspunsuri negative pentru toţi celelalţi vectori. Algoritmul de instruire pentru reţeaua concurentă este următorul:

Etapa 1. *Se creează baza de date. Aceasta conţine vectorii de instruire obţinuţi în urma preprocesării semnalului vocal.*

Etapa 2. *Se extrag din baza de date seturile de vectori specifice fiecărei reţele neurale în parte. Dacă este cazul, se stabilesc ieşirile dorite.*

Etapa 3. *Se aplică algoritmul de instruire fiecărei reţele neurale folosind seturile de vectori create la pasul 2.*

Etapa de recunoaştere se desfăşoară în paralel, conform schemei din figura 4.2.

Cele n reţele neurale se presupune că au fost instruite prin algoritmul descris mai înainte. La aplicarea vectorului de test, reţelele generează câte un răspuns individual, iar selecţia constă în alegerea reţelei care a generat răspunsul cel mai puternic. Reţeaua selectată prin regula *câştigătorul ia tot* este declarată câştigătoare. Indicele reţelei câştigătoare va fi indicele clasei în care este plasat vectorul de test. Această metodă de recunoaştere a formelor presupune, aşadar,

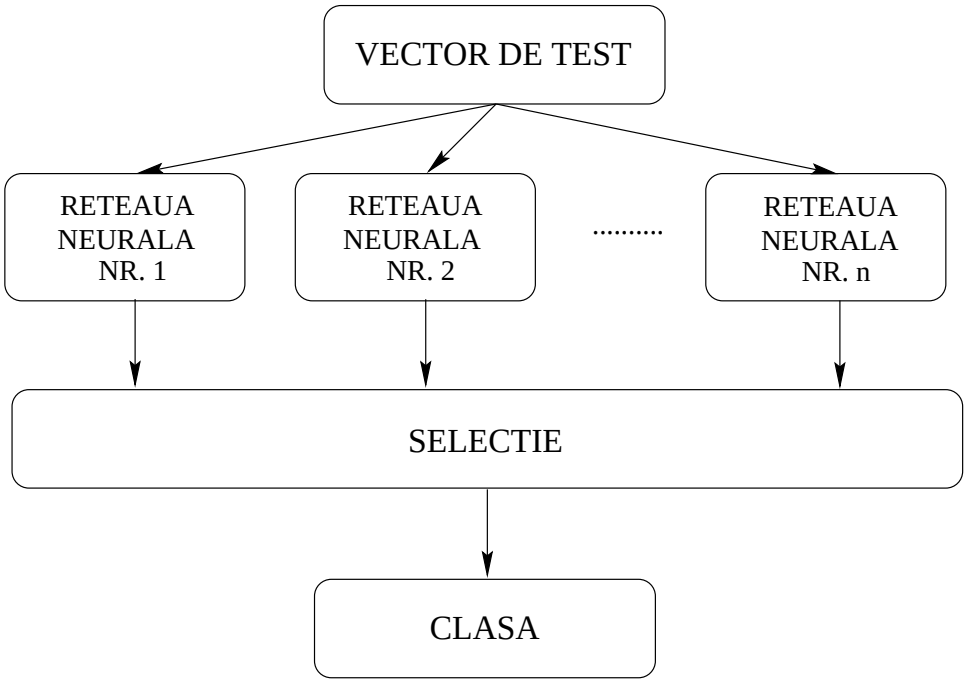

Figura 4.2: Modelul folosit la recunoaştere de reţelele neurale concurente.

că este *a priori* cunoscut numărul de clase cu care va lucra reţeaua concurentă şi că pentru fiecare clasă există suficienţi vectori de instruire. Algoritmul de recunoaştere este cel de mai jos:

Etapa 1. *Este creat vectorul de test prin preprocesarea semnalului vocal.*

Etapa 2. *Vectorul de test este transmis în paralel tuturor celor n reţele neurale instruite în prealabil.*

Etapa 3. *Blocul de selecţie stabileşte indicele reţelei cu răspunsul cel mai bun. Acesta va fi indicele clasei în care este încadrat vectorul.*

Modelele din figurile 4.1 şi 4.2 pot fi particularizate plasând în locul celor n reţele neurale diferite arhitecturi. În experimentele prezentate în această lucrare am folosit perceptroni multistrat (MLP), reţele neurale cu întârziere în timp (TDNN) şi hărţi Kohonen (SOM), obţinând astfel trei tipuri diferite de reţele neurale concurente.

Pentru cele care folosesc MLP şi TDNN se utilizează algoritmul de instruire *backpropagation of error*. Atunci când implementăm reţele neurale concurente cu SOM, abordăm instruirea uşor diferit din cauză că instruirea modulelor

este nesupervizată. De aceea, diferă modul în care alcătuim seturile de vectori pentru instruire, dar şi maniera în care se face selecţia reţelei câştigătoare.

Pentru primele două tipuri de reţea concurentă, MLP-CNN şi TDNN-CNN, am folosit seturi de vectori de instruire cu exemple pozitive şi negative. Un *exemplu pozitiv* pentru o reţea componentă a CNN este un vector din clasa asociată ei. Un *exemplu negativ* este un vector care aparţine altei clase. Pentru a ne asigura că reţelele vor fi capabile să facă diferenţa între exemplele pozitive şi cele negative, am echilibrat numeric seturile de vectori: am inclus un număr egal de vectori pozitivi şi negativi, iar exemplele negative au fost selectate din toate clasele în mod egal.

MLP şi TDNN sunt reţele neurale care sunt instruite supervizat, motiv pentru care fiecărui exemplu pozitiv folosit la instruire i se asociază câte un răspuns dorit. Acesta este un vector n-dimensional, n fiind şi numărul de clase. Fiecare componentă a sa va avea valoarea 1 sau 0. Există o singură componentă 1, plasată pe poziţia care are ca indice numărul clasei din care face parte exemplul pozitiv. Pentru exemplele negative, răspunsul dorit este complementul binar al vectorului corespunzător exemplului pozitiv.

Decizia la recunoaştere se ia pe baza regulii *câştigătorul ia tot*. Fiecare dintre cele n reţele concurente produce în mod independent câte un răspuns la stimulul de test. Se selectează vectorul care conţine exact o componentă apropiată de valoarea 1 şi toate celelalte componente apropiate de valoarea 0, în limitele unei erori acceptabile. Pentru ca răspunsul să fie considerat valid, poziţia componentei maxime în vectorul de ieşire trebuie să coincidă cu indicele reţelei care l-a generat.

Reţeaua concurentă care are la bază SOM este instruită cu seturi de vectori alcătuite doar din exemple pozitive. Ca şi în celelalte cazuri, fiecare reţea componentă este instruită separat, cu setul de vectori special alcătuit pentru ea.

Clasificarea se face prin selecţia hărţii care generează o eroare de cuantizare minimă.

4.3 Recunoaşterea vorbitorilor cu reţele neurale concurente. Experimente

Experimentele descrise în această secţiune [11], [13] compară capacitatea de recunoaştere a MLP, TDNN, SOM şi a reţelelor neurale concurente care folosesc aceste tipuri de reţele neurale drept module de bază. Rezultatele prezentate aici au fost obţinute în urma testelor de recunoaştere a vorbitorilor folosind mai întâi

baza de date Speechdata, iar apoi baza de date Telephdata care conţin cuvinte pronunţate de 25 de vorbitori pe care i-am organizat în 25 de clase.

Scopul testelor este acela de a recunoaşte cei 25 de vorbitori pe baza diverselor pronunţii ale aceloraşi cuvinte, diferite de cele folosite la instruire. Am adoptat ca bază de comparaţie rezultatele obţinute cu MLP, TDNN şi SOM descrise în secţiunea 3.3.2 şi care se găsesc şi în tabelele 4.1, 4.2 şi 4.3 pe coloanele marcate cu litera B (model neural de bază).

Modelul concurent ales de noi grupează 25 de reţele clasice, câte una pentru fiecare vorbitor. Metoda de instruire este descrisă în continuare, fiind supervizată pentru MLP şi TDNN şi nesupervizată pentru SOM. Etapa de recunoaştere impune alegerea celei mai performante reţele care este declarată câştigătoare. Setul de experimente descrise în acest subcapitol testează modelul reţelelor neurale concurente pentru recunoaşterea vorbitorilor.

Tabelul 4.1: Rezultatele experimentelor de recunoaştere a vorbitorilor cu MLP şi MLP-CNN.

MLP			
Speechdata		Telephdata	
B	CNN	B	CNN
71,25%	86,22%	6,75%	72,86%

Tabelul 4.2: Rezultatele experimentelor de recunoaştere a vorbitorilor cu TDNN şi TDNN-CNN.

TDNN			
Speechdata		Telephdata	
B	CNN	B	CNN
57,28%	89,31%	32,33%	72,72%

Tabelul 4.3: Rezultatele experimentelor de recunoaştere a vorbitorilor cu SOM şi SOM-CNN.

SOM			
Speechdata		Telephdata	
B	CNN	B	CNN
52,86%	90,42%	44,72%	75,25%

Am folosit mai întâi MLP şi TDNN drept componente de bază ale CNN.

Fiecărei clase i-am asociat câte una din cele 25 de reţele neurale compuse dintr-un strat de intrare cu 270 de neuroni, două straturi ascunse şi un strat de ieşire alcătuit din 25 de neuroni. Am folosit algoritmul *backpropagation of error* pentru instruirea acestor reţele neurale feedforward şi pentru fiecare dintre ele am creat seturi distincte de vectori de instruire. Astfel, vectorii din clasa 1, cele extrase din cuvintele pronunţate de vorbitorul 1, au fost exemple pozitive pentru reţeaua neurală cu indicele 1 şi exemple negative pentru celelalte 24 de reţele neurale. Răspunsurile dorite asociate exemplelor pozitive au acelaşi format cu cel descris în secţiunea 3.3 unde am discutat despre instruirea MLP şi TDNN ca reţele complexe care recunosc toţi vorbitorii. Am asociat răspunsul dorit $[1\ 0\ 0\ ...\ 0]^t$ exemplelor pozitive pentru reţeaua neurală 1 şi răspunsul dorit $[0\ 1\ 1\ ...\ 1]^t$ exemplelor negative pentru aceeaşi reţea. Pentru a echilibra numărul exemplelor pozitive şi negative, am repetat de 24 de ori exemplele pozitive, făcând această operaţie pentru fiecare set de vectori. Urmărind această idee, am creat seturile de vectori pentru fiecare reţea neurală componentă a MLP-CNN şi TDNN-CNN. În faza de recunoaştere, un vector de test trebuie să activeze doar o singură reţea neurală, cu alte cuvinte, acea reţea neurală activă trebuie să producă un răspuns apropiat de cel al unui exemplu pozitiv. Toate celelalte reţele neurale componente ale CNN trebuie sa genereze un răspuns apropiat de răspunsul unui exemplu negativ. Luăm o decizie asupra recunoaşterii numai atunci când exact o reţea neurală dintre cele 25 generează un răspuns pozitiv şi toate celelalte generează răspunsuri negative. În caz contrar, nu se poate stabili apartenenţa vectorului de intrare la nici o clasă.

Figura 4.3 prezintă mediile ratelor de recunoaştere ale fiecărei reţele neurale compunentă a CNN pentru 25 de vorbitori în două experimente care ilustrează folosirea TDNN şi TDNN-CNN cu baza de date Telephdata. Este evidentă diferenţa dintre mediile ratelor de recunoaştere deşi, aşa cum se observă, este posibil ca TDNN să dea rezultate mai bune pentru anumiţi vorbitori.

Instruirea SOM-CNN a fost mult mai simplă. Am creat 25 de seturi de vectori şi am folosit algoritmul nesupervizat de instruire a reţelelor Kohonen pentru instruirea independentă a fiecăreia dintre cele 25 de reţele Kohonen. De această dată, am folosit doar exemple pozitive. La recunoaştere, vectorul de test este aplicat în paralel tuturor SOM componente ale SOM-CNN. Reţeaua în care apare cea mai mică eroare de cuantizare este declarată câştigătoare, iar indicele său este indicele clasei din care se presupune că face parte vectorul de test.

În tabelele 4.1, 4.2 şi 4.3 sunt prezentate ratele de recunoaştere pentru MLP-CNN, TDNN-CNN şi SOM-CNN, în coloanele marcate cu CNN.

Rezultatele obţinute la recunoaşterea vorbitorilor cu reţele neurale con-

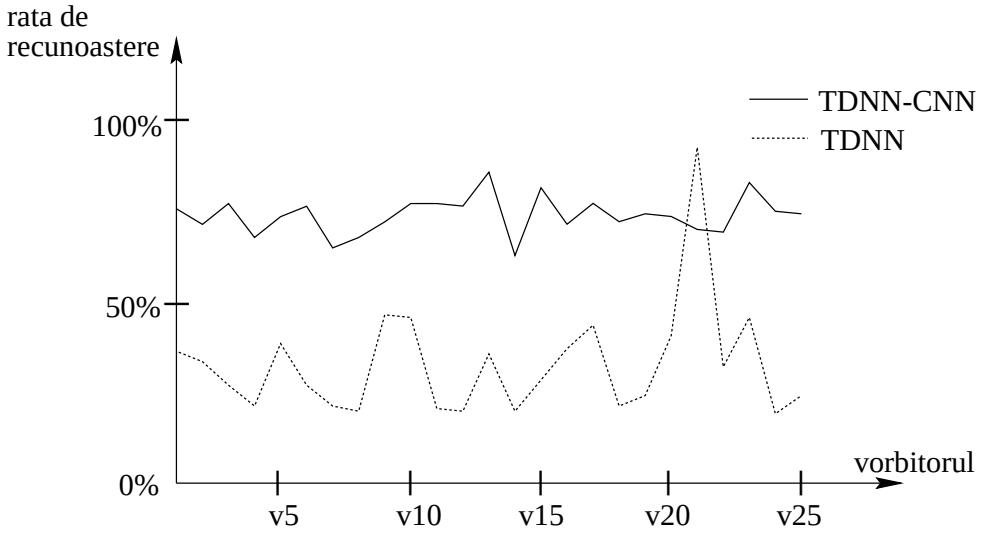

Figura 4.3: Mediile ratelor de recunoaştere pentru cei 25 de vorbitori în două experimente care ilustrează folosirea TDNN şi TDNN-CNN cu Telephdata.

curente sunt mult îmbunătăţite faţă de modelele clasice. MLP s-a dovedit a fi sensibil mai bun decât TDNN şi SOM în testele cu modelul clasic folosind Speechdata, dar mult mai slab pentru Telephdata. Folosirea reţelelor concurente a îmbunătăţit semnificativ ratele de recunoaştere. Modelul SOM-CNN s-a dovedit a fi cel mai performant, rezultatele fiind cu mai mult de 4% mai bune decât modelul MLP-CNN şi cu aproximativ 1% mai bune decât modelul TDNN-CNN. În general, s-a păstrat o diferenţă de aproximativ 15% între testele cu Speechdata şi cele cu Telephdata. Experimentele au arătat că folosind câte o reţea specializată pentru fiecare cuvânt se obţin rezultate mai bune la recunoaştere, chiar dacă faza de instruire durează în mod evident mai mult.

Este cunoscut faptul că algoritmul *backpropagation of error* are o convergenţă lentă, iar atunci când condiţia de oprire înseamnă o eroare de clasificare mică, instruirea poate să fie foarte costisitoare din punct de vedere al timpului alocat. Când instruim un set de reţele neurale concurente, timpul de instruire este proporţional cu numărul de module. Odată instruite aceste reţele neurale, timpul de calcul pentru stabilirea apartenenţei unui vector de intrare la o clasă devine foarte scurt deoarece constă dintr-o singură propagare a informaţie dinspre intrări înspre ieşiri. În experimentele noastre, am alocat perioade până la 24 de ore pentru o instruire foarte fină a tuturor modulelor. Algoritmul de instruire a hărţilor cu autoorganizare este mai rapid deoarece condiţia de terminare este dată de numărul de paşi de execuţie. Faza de recunoaştere este, însă, mai lentă decât pentru reţelele neurale feedforward. Pentru stabilirea erorii minime de cuantizare, aceasta presupune compararea fiecărui vector de

test cu toate prototipurile.

Pentru experimentele de recunoaştere a vorbitorilor cu MLP-CNN şi TDNN-CNN am folosit module instruite prin programul Stuttgart Neural Network Simulator (SNNS). Rezultatele modulelor au fost combinate printr-o aplicaţie pe care am scris-o în limbajul de programare oferit de SNNS. Pentru testele de recunoaştere cu SOM-CNN am instruit hărţile cu ajutorul programelor din pachetul SOM_PAK, iar pentru combinarea rezultatelor am realizat o aplicaţie în limbajul de programare C++.

4.4 Recunoaşterea vocalelor cu reţele neurale concurente

După ce am văzut că modelul CNN aduce îmbunătăţiri remarcabile ratelor de recunoaştere a vorbitorilor în comparaţie cu modelele clasice, în această secţiune ne vom concentra asupra recunoaşterii unor vocale din limba română având la dispoziţie diverse variante de pronunţie a cifrelor de la 0 la 9. Înainte de a analiza metoda implementată şi rezultatele acestor experimente, vom face o prezentare a principalelor noţiuni de fonetică şi fonologie specifice limbii române [3].

4.4.1 Fonetică şi fonologie în limba română

Vorbirea este un mijloc de transmitere orală a mesajelor şi presupune existenţa unui sistem de sunete şi a unui set de reguli de asociere a lor. Sunetele limbii alcătuiesc unităţile de bază ale acesteia, fiind acceptate în limitele unor variante şi varietăţi admisibile.

Elemente de fonetică

Sunetele fonice articulate sunt unităţile minimale purtătoare de informaţie ale unui limbaj vorbit, stând la baza comunicării dintre oameni. Ele constau în vibraţii ale aerului, modulate de organele vorbirii. Sunt caracterizate prin tărie, înălţime, timbru, durată, trăsături perceptibile de urechea umană. *Fonetica* este disciplina care studiază producerea, transmiterea şi receptarea sunetelor limbajului articulat, adică a sunetelor vorbite.

Sunetele fonice articulate sunt produse la intrarea în vibraţie a aerului expirat. La acest proces participă plămânii, traheea, laringele, coardele vocale, glota, epiglota, faringele, maxilarele, gura, buzele, dinţii, limba, fosele nazale. Ansamblul organelor care participă la producerea sunetelor se numeşte *aparat fonoarticulator*.

Articularea este produsă în timpul expiraţiei, adică în momentul evacuării aerului din plămâni. În acest fel ia naştere fluxul de aer care produce unda fonică articulată. Această cantitate de aer trece printr-o serie de organe şi cavităţi care, în principal, participă la funcţia biologică a respiraţiei şi care formează *aparatul vorbirii*. În interiorul laringelui se află două lamele formate din ţesuturi elastice şi muşchi numite *coarde vocale* care delimitează în interiorul lor un spaţiu numit *glota*. Coardele vocale realizează mişcări de închidere şi deschidere a glotei. În funcţie de acest grad de închidere, dar şi de viteza cu care aerul este evacuat din plămâni, apar sunete mai înalte sau mai joase şi de amplitudini diferite. Glota este un organ complex al vorbirii care produce prima diferenţiere a sunetelor. Astfel, sunetele vor fi *vocoide* sau *contoide sonore* ori *surde*.

Faringele, *cavitatea bucală* şi cea *nazală* sunt rezonatori în care sunetul este articulat, modulat şi amplificat. *Uvula* dirijează fluxul fonic înspre cavitatea bucală sau nazală determinând *rezonanţa bucală* sau *rezonanţa nazală*. *Limba* este organul cu rolul cel mai important în articulare, iar *buzele* formează un ultim rezonator de îngustare sau ocluziune a fluxului fonic.

Articularea reprezintă totalitatea mişcărilor organelor vorbirii care determină forma şi mişcările rezonatorilor, cu influenţe asupra formării, modificării şi amplificării fluxului fonic. Articularea are trei componente:

- *modul de articulare*, determinat de forma şi mişcările rezonatorilor;
- *locul de articulare*, adică organul articulării;
- *prezenţa sau absenţa vibraţiilor* glotale.

Caracteristica acestor componente este că ele pot apărea doar în combinaţie, şi nu izolate una de alta. Prin combinaţia lor se pot defini tipurile articulatorii care pot apărea în limba română vorbită.

Vocalele (*vocoidele*) sunt sunete produse de coloana de aer vibrat prin trecerea ei liberă prin traiectul vocal. Diversele vocale rezultă prin gradul de deschidere a rezonatorului bucal combinat cu mişcările verticale şi orizontale ale limbii. Ca mod de articulare, vocalele pot fi *deschise* (*a*), *semiînchise* (*e, ă, o*) şi *închise* (*i,î, u*), în funcţie de nivelul de apropiere a limbii de palatul moale şi de gradul de deschidere a maxilarelor. După locul de articulare, ele pot fi *posterioare* (*u, o*), *centrale* (*ă, î*) sau *anterioare* (*e, i*), iar *a* este o vocală neutră. Pe lânga muşchiul lingual, un rol important îl joacă poziţia buzelor, producând o vocală *rotunjită* (*o, u*) sau *nerotunjită* (*a, e, i, ă, î*). Conform acestor criterii, sistemul vocalic românesc se poate sintetiza ca în tabelul 4.4.

Consoanele sunt produse de fluxul de aer care întâlneşte în calea sa diferite obstacole. Diferenţele între ele se fac încă de la nivelul rezonatorului glotal,

Tabelul 4.4: Sistemul vocalic al limbii române.

Poziţia buzelor		nerotunjite		rotunjite
Poziţia limbii		anterioare	centrale	posterioare
Deschiderea gurii	deschise	a		
	semiînchise	e	ă	o
	închise	i	î	u

consoanele fiind sonore sau surde. Perechile de consoane clasificate dupa criteriul sonore – surde pot fi grupate în felul următor: *b–p, d–t, v–f, z–s, ž–ts, ğ–č, g'–k', g–k.* Toate aceste consoane sunt *nesonante.* În prezenţa vibraţiilor faringiene, apar consoanele *sonante, l, m, n, r.* După locul de articulare, consoanele se clasifică astfel: *bilabiale: b, p, m* (buze), *labiodentale: v, f* (buza inferioară şi dinţii superiori), *dentale: d, t, z, s, ts, n, l, r* (dinţii), *prepalatale: ž, š, ğ, č* (palatul anterior), *palatale: g', k'* (palatul mediu), *velare: g, k* (palatul moale) şi *laringala h* (laringele). La producerea unor constante, palatul moale permite trecerea liberă a unei părţi din fluxul de aer fonator prin fosele nazale, producându-se consoanele *nazale: m, n.* Alteori, traiectul vocal este închis complet. Prin deschiderea lui bruscă se produc consoanele *oclusive (explozive): b, p, m, d, t, n, g', k', g, k.* Când traiectul vocal este doar îngustat şi aerul îl parcurge tot timpul cât durează emisia, se produc consoanele *constrictive.* Acestea sunt *fricative: v, f, z, s, ž, š,* când îngustarea este mai pronunţată, sau *lichide: l, r.*

Consoana *r* este *vibrantă* pentru că vârful limbii vibrează, iar *l* este *laterală* deoarece aerul se scurge de-a lungul marginilor limbii. Când traiectul vocal este iniţial închis complet şi se deschide treptat, se produc consoanele *semioclusive (africate): ts, č, ğ.* Sistemul consonantic al limbii române este alcătuit din 22 de consoane care se clasifică aşa cum se vede în tabelul 4.5.

Semivocalele sunt articulate ca şi vocalele, dar durata lor este mai mică şi efortul respirator mai mare. Limba română are următoarele semivocale: *e̲, i̲, o̲, u̲.*

Elemente de fonologie

Fonemele sunt unităţi minimale ale sistemului limbii, caracterizate prin trăsături distinctive a căror modificare ar altera chiar sistemul limbii [33].

Tabelul 4.5: Sistemul consonantic al limbii române.

Vibraţiile glotale		sonante			nesonante		
					surde/sonore	surde/sonore	surde/sonore
Modul de articulare		oclusive nazale	constrictive		explozive	fricative	africate
			vibrante	laterale			
Locul de articulare	bilabiale	m	-	-	b p	- -	- -
	labiodentale	-	-	-	- -	v f	- -
	dentale	n	r	l	d t	z s	- ts
	prepalatale	-	-	-	- -	ž š	ğ č
	palatale	-	-	-	g' k'	- -	- -
	velare	-	-	-	g k	- -	- -
	laringale	-	-	-	- -	- h	- -

Fonemul este o entitate care nu poate fi analizată în unităţi mai mici şi are funcţia de a diferenţia între ele cuvinte diferite sau forme gramaticale ale aceluiaşi cuvânt. *Fonologia* studiază funcţia fonemelor într-o limbă şi stabileşte sistemul de foneme al acelei limbi, realizând inventarierea lor.

Fonemul este o clasă de sunete asemănătoare, cu aceeaşi funcţie. În limba vorbită, un fonem poate fi realizat prin mai multe *variante* care sunt sunete asemănătoare din punct de vedere al pronunţiei şi al percepţiei, dar diferă datorită contextelor fonetice în care se găsesc. De asemenea, un fonem poate avea mai multe *varietăţi* care sunt pronunţii ale aceluiaşi sunet de către vorbitori diferiţi [3]. Aşadar, fonemul este un element abstract şi cu caracter general, iar sunetul este o particularizare, o concretizare a fonemului.

Fonemul are două funcţii: *distinctivă* şi *contrastivă*. Prima funcţie, cea distinctivă, realizează distingerea cuvintelor şi a formelor lor (**d**ar–**f**ar–**v**ar–**z**ar). Există mai multe grupe de foneme care *comută*: *d, f, v, z; p, r, t; a, o, u; u, i; k, č*. Fiecare fonem are propriile trăsături relevante legate de modul şi locul de articulare, dar şi de prezenţa sau absenţa vibraţiilor glotale. Aceste trăsături le deosebesc unele de altele şi creează opoziţiile între foneme. Cea de-a doua funcţie, cea contrastivă permite stabilirea structurii fonematice a unui cuvânt, facilitând înţelegerea lui.

Cele şapte vocale din limba română pot constitui centrul de sonoritate al unei silabe sau pot alcătui singure o silabă, adică sunt *silabice*. Semivocalele se definesc drept vocale nesilabice şi apar în structura *diftongilor* sau a *triftongilor*. Diftongii sunt formaţi dintr-o vocală şi o semivocală, iar triftongii din două semivocale şi o vocală. Diftongii şi triftongii sunt segmente vocalice constituite din două, respectiv trei vocale din care doar una se realizează ca vocală. Consoanele din sistemul consonantic al limbii române sunt *nesilabice*, fiind dependente de o vocală.

Silaba este construcţia superioară fonemului şi este caracterizată printr-un singur accent. Ea este formată dintr-un singur fonem care trebuie să fie neapărat o vocală sau din mai multe foneme între care se găseşte o vocală.

Accentul evidenţiază o silabă printr-o pronunţare mai intensă şi este în contrast cu silabele neaccentuate.

4.4.2 Experimente de recunoaştere a vocalelor

Abordăm problema recunoaşterii vocalelor prin studierea similarităţii dintre vectorii prototip şi vectorii obţinuţi prin analiza cepstrală a regiunilor vocalizate din cuvintele ţintă. Prezentăm în această secţiune un sistem alcătuit din reţele neurale concurente pentru recunoaşterea vocalelor care apar în pronunţiile în

limba română ale cifrelor de la 0 la 9.

Analiza cepstrală a semnalelor vocale

Cepstrul se obţine printr-o transformare complexă care se poate aplica şi asupra semnalului vocal. Această transformare produce separarea unor componente ale semnalului care apoi sunt combinate liniar. Pentru semnalul:

$$s(n),\ n = 0, 1, 2 \ldots,$$

transformata sa va fi de forma:

$$c_s(n) = c_e(n) + c_\Theta(n),\ n = 0, 1, 2 \ldots$$

În această relaţie, $c_e(n)$ este partea din cepstru care reprezintă contribuţia excitaţiei periodice, iar $c_\Theta(n)$ este partea corespunzătoare funcţiei de transfer a traiectului vocal. Ca urmare a acestei transformări, partea inferioară a cepstrului unui semnal vocal, pentru care n are valori mici, este asociată semnalului $c_\Theta(n)$, iar partea superioară, în care n este mare, corespunde semnalului $c_e(n)$.

Baza de date

Am preprocesat prin analiza cepstrală bazată pe scala mel pronunţiile în limba româna ale celor 10 cifre. Fiecare dintre cei 4 subiecţi a pronunţat aceste cuvinte de câte 10 ori. De fiecare dată am extras vectorii de instruire a sistemului de recunoaştere din primele 5 repetiţii, păstrând ultimele 5 repetiţii pentru teste. Am avut astfel la dispoziţie 200 de repetiţii pentru antrenarea sistemului şi alte 200 pentru verificarea sa. Pentru instruire, am segmentat manual fiecare repetiţie şi am selectat regiunile din semnal corespunzătoare vocalelor şi semivocalelor. Extragerea vectorilor de caracteristici a fost realizată prin parcurgerea semnalelor selectate cu o fereastră de 256 de eşantioane şi un pas de 50 de eşantioane. Din fiecare fereastră s-au calculat câte 12 coeficienţi cepstrali care formează vectorul asociat ei. Numărul de ferestre şi, implicit, numărul de vectori extraşi din semnalul asociat unui fonem este influenţat de lungimea acestuia în timp, adică de caracterul vocalic sau semivocalic şi de debitul verbal, ca şi de modul de pronunţie specific fiecărui vorbitor. La nivelul întregii colecţii de pronunţii, numărul de vectori asociaţi unui fonem este direct proporţional cu frecvenţa de apariţie a vocalelor în cele 10 cuvinte. Cuvintele au fost înregistrate în condiţii normale, fără o amenajare specială a incintei, cu un microfon de uz general. Tabelul 4.6 conţine o statistică a numărului de vectori pe care i-am obţinut pentru fiecare fonem din colecţia de semnale vocale descrisă mai sus. Figura 4.4 prezintă câte o secvenţa tipică de coeficienţi

cepstrali calculaţi pentru fiecare dintre cele şase vocale care alcătuiesc baza de date pentru un vorbitor, iar în anexa A am listat valorile numerice pe baza cărora am trasat aceste grafice.

Se observă că în partea inferioară a cepstrului apar diferenţe vizibile între secvenţele asociate acestor foneme, confirmând astfel că traiectul vocal contribuie într-o măsură importantă la producerea fonemelor. În a doua jumătate a cepstrului, diferenţele nu mai sunt atât de evidente, însă se remarcă un vârf datorat frecvenţei fundamentale a semnalului excitator.

Energia semnalului vocal

Analiza oricărui semnal se face într-un timp finit. Pentru aceasta trebuie să îi cunoaştem caracteristicile pe toată durata şi proprietatea pe care o căutăm trebuie să rămână invariantă. Semnalul vocal se caracterizează prin faptul că nu este staţionar pentru un timp infinit de lung. Proprietatea de staţionaritate rămâne valabilă doar pentru câteva milisecunde. Acesta este motivul pentru care metodele de analiză "pe termen lung" a semnalelor sunt aplicate în contextul nostru pe o fereastră. Analiza pe fereastră presupune "decuparea" unei regiuni din semnal şi prelucrarea sa în mod independent. Vorbirea este un proces dinamic, de aceea un singur cadru nu este întotdeauna suficient. Din necesitatea de a surprinde caracteristicile temporale ale semnalului vocal, obişnuim să folosim cadre extrase cu ajutorul unei ferestre care se deplasează de-a lungul acestuia.

Atunci când dorim să extragem informaţia pe care ne-o oferă cadrul care cuprinde porţiunea de semnal vocal din intervalul $n = m - N + 1, \ldots, m$ folosim o versiune adaptată a metodei care se foloseşte de regulă pentru semnale infinite în timp [16].

Să presupunem că dorim să ştim dacă o secvenţă de semnal este vocalizată sau nevocalizată pe porţiunea care se termină la momentul $n = m$. Se cunoaşte că semnalul vocalizat are, în general, o energie mai mare decât semnalul nevocalizat. Folosim această idee pentru a ne asista în luarea deciziei. Având două semnale $s_1(n)$ vocalizat şi $s_2(n)$ nevocalizat de durată infinită, energia lor va fi în relaţia:

$$E_{s_1} > E_{s_2},$$

unde

$$E_s \stackrel{\Delta}{=} \sum_{n=-\infty}^{\infty} s^2(n).$$

Pentru o fereastră de semnal care conţine punctele din jurul lui m avem inega-

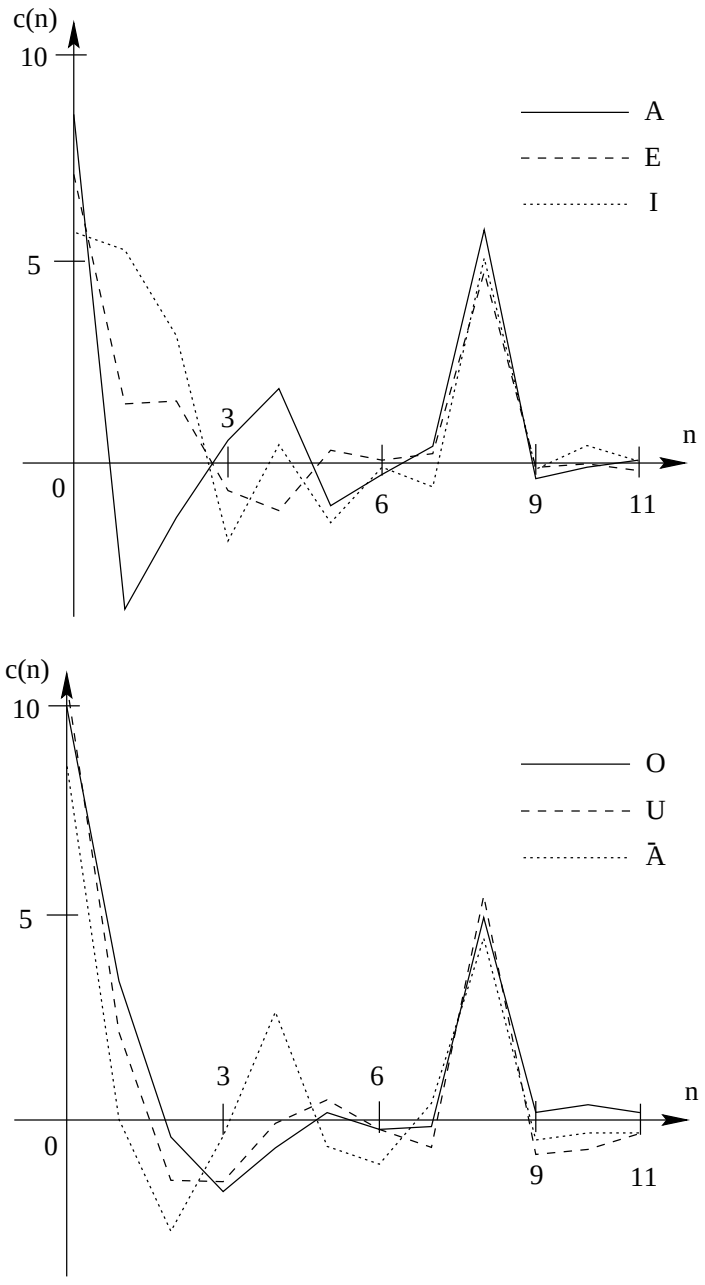

Figura 4.4: Secvenţe tipice de coeficienţi cepstrali asociate fonemelor A, E, I, O, U, Ă.

litatea:

$$E_{s_1}(m) > E_{s_2}(m),$$

unde

$$E_s(m) = \sum_{n=m-N+1}^{m} s^2(n).$$

În această relaţie, N este numărul eşantioanelor din semnal care se regăsesc în interiorul ferestrei. Ca să putem automatiza luarea unei decizii privind caracterul vocalizat sau nevocalizat al semnalului, raportăm valoarea lui $E_s(m)$ la un prag: dacă valoarea energiei depăşeşte pragul semnalul este vocalizat, iar dacă valoarea sa este sub prag, semnalul este nevocalizat.

Un element important în extragerea unor informaţii pe baza energiei este fixarea numărului N de eşantioane cuprinse în fereastră. Atunci când fereastra are dimensiune mare, graficul energiei obţinut prin baleierea semnalului este mai "neted". Aceasta înseamnă că în vecinătatea pragului decizia nu este influenţată de "vârfuri" ale energiei. Pe de altă parte, o fereastră de dimensiuni mari poate provoca "netezirea" prea pronunţată a formei energiei. Drept consecinţă, fie zona vocalizată determinată prin calcul depăşeşte dimensiunile reale, fie nu se mai pot detecta zonele nevocalizate de durată scurtă inserate între două zone vocalizate (fig. 4.5).

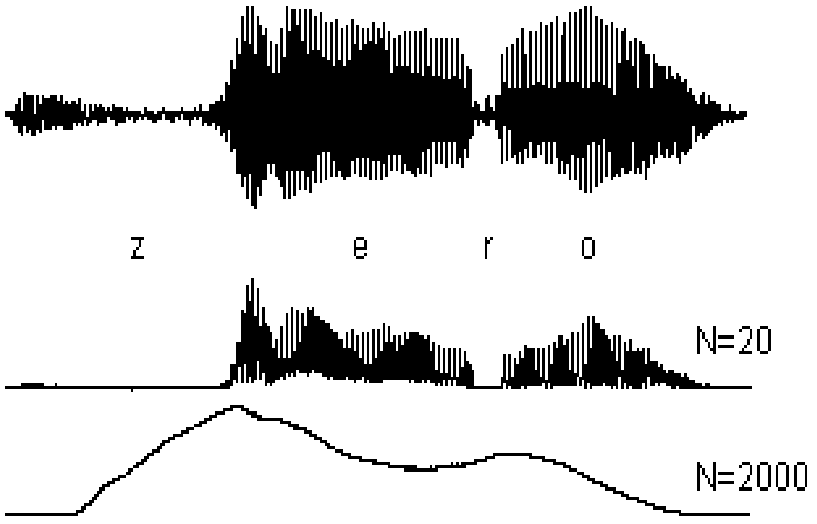

Figura 4.5: Funcţia de energie a semnalului corespunzător unei pronunţii a cifrei *zero* pentru N=20 şi N=2000.

Datele pe care le obţinem prin studierea energiei au un caracter preponderent informativ şi nu influenţează în mod fundamental corectitudinea recunoaşterii vocalelor din semnal. De aceea, în scopul simplificării calculelor

folosim o fereastră dreptunghiulară a cărei dimensiune am stabilit-o experimental la $N = 300$. Deoarece am eşantionat semnalul vocal cu o frecventă de 8000 Hz, fereastra folosită la calculul energiei selectează o porţiune de 37,5 ms. Am ales o fereastră dreptunghiulară pentru că ne permite calculul recurent al energiei:

$$E_s(m) = E_s(m-1) + s^2(m) - s^2(m-N).$$

Astfel, timpul de estimare a energiei semnalului este liniar, fiind proporţional doar cu dimensiunea semnalului vocal şi nu depinde de dimensiunea ferestrei.

Pentru a lua o decizie în legătură cu caracterul vocalizat sau nevocalizat al semnalului într-o anumită regiune, am stabilit două praguri pentru energie: unul superior şi unul inferior. Atunci când curba ascendentă a energiei depăşeşte pragul superior, considerăm că începe o zonă vocalizată. Când pentru un segment vocalizat energia scade sub pragul inferior, considerăm că începe o zonă nevocalizată. Avantajele folosirii celor două praguri sunt, pe de o parte, o mai bună delimitare a începutului zonei vocalizate şi, pe de altă parte, evitarea plasării în zona nevocalizată a vocalelor mai puţin accentuate de la sfârşitul cuvintelor.

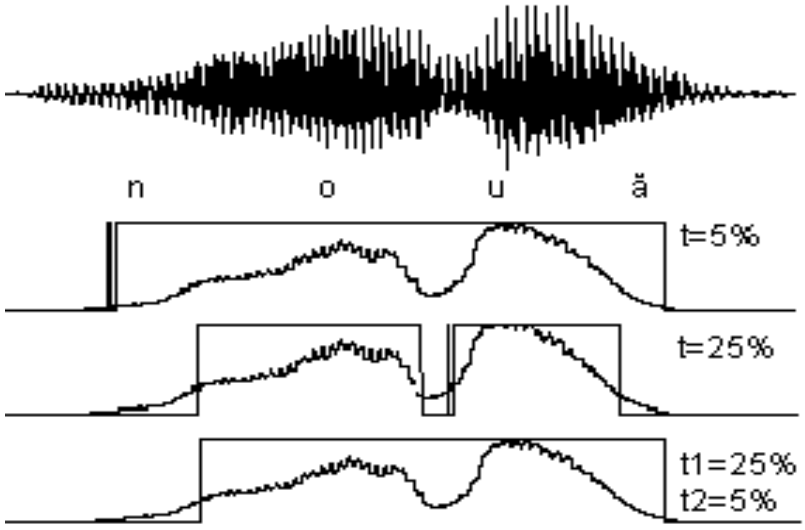

Figura 4.6: Stabilirea zonelor vocalizate şi a celor nevocalizate pentru pronunţia cifrei *nouă*. Valorile pragurilor sunt raportate la nivelul maxim atins de energie.

Figura 4.6 prezintă modul în care se decide dacă o zonă este vocalizată sau nu atunci când se foloseşte un prag şi când se folosesc două praguri, pentru o pronunţie a cifrei *nouă*. Atunci când pragul unic are valoare mică, regiunea vocalizată cuprinde şi fonemul *ă*. Această zonă cuprinde în mod eronat şi fonemul *n* care, deşi este nevocalizat, are amplitudine mare în comparaţie cu cea a unei vocale. Un prag mare elimină fonemul *n*, dar ignoră şi fonemul *ă*

care datorită poziţiei sale finale are o amplitudine mică. Pragul unic are în plus dezavantajul că zona vocalizată poate debuta şi se poate încheia cu o oscilaţie, comportament eliminat de folosirea a două praguri.

Analiza semnalului pe baza energiei dă o primă informaţie legată de prezenţa sau absenţa zonelor vocalizate. Pe baza ei restrângem domeniul de căutare. A doua etapă constă în stabilirea identităţii fiecărei vocale în interiorul regiunilor vocalizate.

Recunoaşterea vocalelor

Ca şi la recunoaşterea vorbitorilor, am realizat un set de experimente folosind mai întâi o singură reţea neurală pentru recunoaşterea tuturor vocalelor, iar apoi CNN. De data aceasta am testat doar reţeaua Kohonen, date fiind rezultatele pe care le-am obţinut la recunoaşterea vorbitorilor şi i-am comparat comportamentul cu cel al unui SOM-CNN.

Pentru varianta care foloseşte un singur SOM am instruit reţeaua cu întreaga secvenţă de vectori obţinuţi după preprocesarea semnalelor vocale selectate. Am instruit în două etape o reţea Kohonen cu 10x15 noduri prin algoritmul SOFM descris în paragraful 2.9.2. Prima etapă, cea de organizare a clusterelor, s-a desfăşurat de-a lungul a 1000 de paşi şi vecinătatea a scăzut treptat până la un singur neuron. În a doua etapă, instruirea s-a făcut în 10000 de paşi, iar vecinătatea a rămas fixată la dimensiunea minimă. În urma instruirii şi a calibrării reţelei neurale cu vectorii de instruire am obţinut un set de prototipuri etichetate a cărui structură este cea din tabelul 4.6.

Tabelul 4.6: Numărul de vectori de instruire a SOM şi SOM-CNN, numărul de prototipuri obţinute în urma instruirii SOM şi numărul de prototipuri obţinute după instruirea SOM-CNN.

	A	E	I	O	U	Ă
Instruire	372	516	339	525	453	52
SOM	42	37	16	21	21	2
SOM-CNN	25	25	25	25	25	25

Se observă că fonemul *ă* este cel mai slab reprezentat din cauza volumului redus al datelor de instruire disponibile. Numărul total de prototipuri este mai mic decât cel al neuronilor din reţea pentru că nu toţi neuronii au fost etichetaţi la calibrare. În figura 4.7 este prezentată proiecţia U-matrix pentru această reţea Kohonen. *Unified distance matrix* (U-matrix) [47] este o metodă de vizualizare printr-o imagine care foloseşte nivele de gri a clusterelor generate

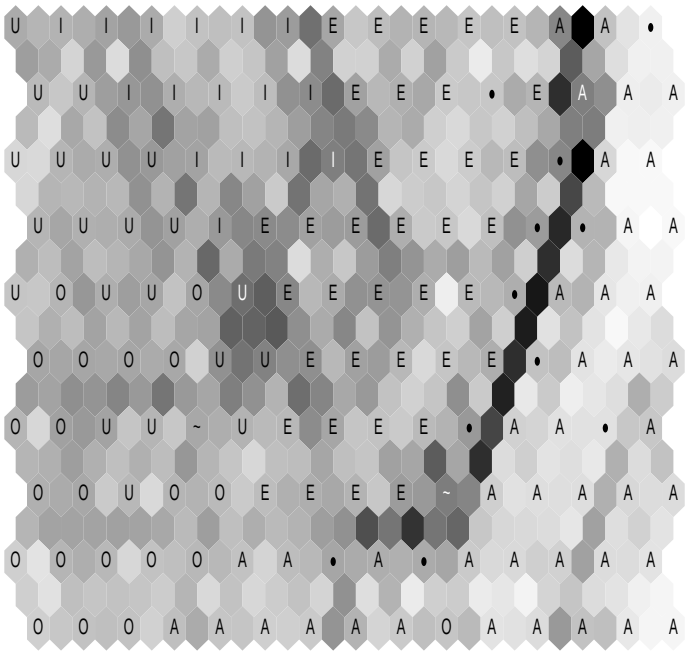

Figura 4.7: U-matrix pentru reţeaua Kohonen instruită să recunoască fonemele *a*, *e*, *i*, *o*, *u*, *ă*. Pentru fonemul *ă* am folosit simbolul $\sim$.

de o reţea Kohonen.

Şi aici se observă slaba reprezentare a prototipurilor fonemului *ă*. Prototipurile celorlalte foneme formează grupuri compacte şi poziţia clusterelor reflectă calităţile acustico-fonetice ale fonemelor.

Instruirea componentelor SOM-CNN s-a făcut după gruparea manuală a vectorilor de instruire în şase clase. Fiecare reţea neurală componentă s-a specializat în recunoaşterea unei singure clase. Am plasat în acelaşi grup atât realizările ca vocale cât şi ca semivocale ale fonemelor. Am folosit şase reţele Kohonen cu câte 5x5 neuroni instruite de asemenea în două etape. Numărul total de neuroni al SOM-CNN este egal cu numărul de neuroni al SOM folosit la experimentul cu o singură reţea neurală. Aceasta înseamnă că numărul prototipurilor ar trebui să fie egal pentru ambele tipuri de experimente: 150. Diferenţa este minoră şi apare datorită faptului că în primul caz nu toţi neuronii au fost etichetaţi la calibrare, aşa cum am arătat în tabelul 4.6.

Forma semnalului corespunzător uneia dintre cele 20 de pronunţii disponibile ale cifrei *nouă* este desenată în figura 4.8. Am delimitat regiunea din semnal corespunzătoare fonemului *ă* pe care am folosit-o la instruire şi din care am extras 4 vectori. În vorbirea curentă pronunţia cuvântului *nouă* este uneori distorsionată şi acesta se transformă în *noo*. În baza noastră de date am avut un astfel de exemplu pe care l-am eliminat şi din acest motiv am folosit doar

Figura 4.8: Regiune selectată pentru instruirea reţelelor Kohonen care recunosc fonemul *ă*. Semnalul este al uneia dintre cele 20 de pronunţii ale cuvântului *nouă*.

19 variante ale cuvântului *nouă*.

Vom descrie în continuare tehnica pe care am aplicat-o la recunoaştere. Zonele vocalizate din semnalul de test sunt preprocesate cu ajutorul unei ferestre din care se calculează câte un vector de componente cepstrale. Fereastra se deplasează cu un pas de 50 de eşantioane şi obţinem o colecţie de vectori a căror succesiune descrie evoluţia semnalului vocal. Pentru fiecare vector reţinem poziţia care îi corespunde în semnal şi eroarea de cuantizare minimă, respectiv eticheta pe care le calculează reţeaua neurală. Experimental am stabilit un prag maxim al erorii de cuantizare pentru a elimina fonemele care se presupune că nu aparţin nici unei clase, de exemplu consoanele. Am obţinut prin acest procedeu o succesiune de etichete de clase care arată cum au fost recunoscute vocalele de către sistem. În figura 4.9 se poate vedea cum a fost recunoscută o repetiţie a cuvântului *nouă* de SOM şi SOM-CNN.

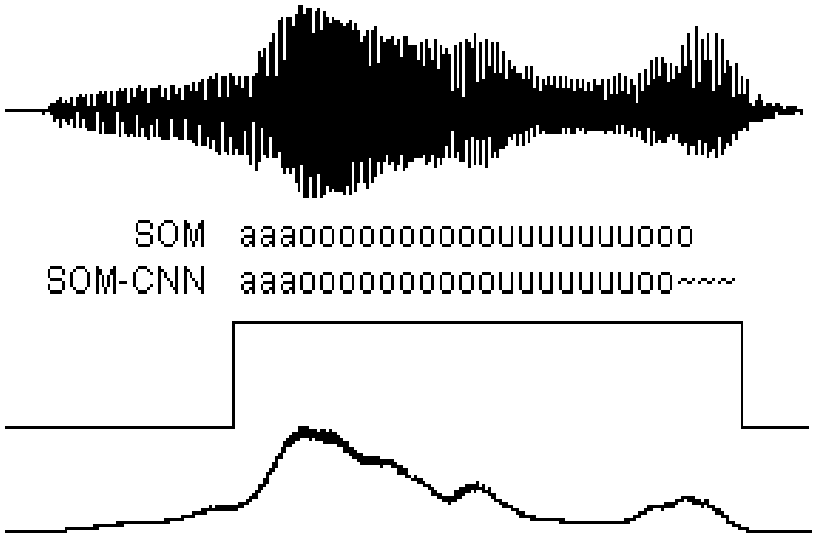

Figura 4.9: Interpretarea cuvântului *nouă* de către SOM şi SON-CNN. Am folosit simbolul ~ pentru fonemul *ă*.

Se remarcă faptul că SOM nu recunoaşte fonemul *ă* final. SOM a decis că

în acea zonă se găseşte doar de fonemul *o*. În vorbirea curentă, trecerea de la *u* la *ă* în cuvântul *nouă* se face prin uşoara inserare între ele a semivocalei *o*, fapt sesizat atât de SOM cât şi de SOM-CNN.

Tabelul 4.7: Rezultatele experimentelor de recunoaştere a vocalelor şi semivocalelor cu SOM şi SOM-CNN.

	SOM	SOM-CNN
E4.1	73,68%	84,21%
E4.2	62,47%	73,93%
E4.3	50%	56,25%
E4.4	5,88%	5,48%
E4.5	13,07%	8,61%

Rezultate experimentale

Pentru a compara performanţele SOM şi SOM-CNN, în tabelul 4.7 sunt descrise rezultatele următoarelor experimente [12]:

- E4.1: stabilirea procentului vocalelor şi semivocalelor recunoscute de cele două modele;
- E4.2: proporţia în care dimensiunea reală a zonei vocalei sau a semivocalei este sesizată de SOM şi SOM-CNN;
- E4.3: numărul de succesiuni vocală-semivocală recunoscute corect;
- E4.4: procentul anunţurilor false în legătură cu existenţa unei vocale sau semivocale într-o regiune din semnal în care în realitate sunt consoane;
- E4.5: procentul confuziilor dintre două vocale.

Experimentul 4.1 ne ajută să ne formăm o imagine corectă asupra capacităţii de a recunoaşte noi vectori de către SOM şi SOM-CNN. Progresul obţinut cu ajutorul modelului nostru este de peste 10%. Am considerat că o vocală este recunoscută corect dacă secvenţa de etichete este cea reală pe zona centrală a segmentului vocalic, aşa cum se poate vedea şi în figura 4.9.

Fenomenul coarticulaţiei influenţează recunoaşterea în zonele de tranziţie dintre două foneme. Prin experimentul 4.2 stabilim în ce măsură sistemele testate de noi determină dimensiunile corecte ale vocalelor şi semivocalelor.

Atunci când apar în diftongi, vocalele sunt pronunţate cu mai multă forţă şi tind să domine semivocalele, aşa cum se întâmplă şi pentru cuvântul *doi*.

Experimentul 4.3 stabileşte procentul de astfel de succesiuni pentru care sunt recunoscute atât vocala cât şi semivocala.

Experimentele 4.4 şi 4.5 determină capacitatea SOM şi a SOM-CNN de a evita confuziile. Energia semnalului vocal dă o primă informaţie referitoare la caracterul vocalizat sau nevocalizat al unei porţiuni din semnal. Reţeaua neurală stabileşte pentru segmentele vocalizate care este succesiunea fonemelor. Avem, astfel, două nivele de decizie a existenţei vocalelor. În experimentul 4.4 am calculat rata confuziilor între vocale şi consoane.

În experimentul 4.5 am stabilit rata confuziilor între vocale, determinând care este procentul în care o vocală este înlocuită de sistemul de recunoaştere cu altă vocală.

4.5 Concluzii

Rezultatele obţinute în urma experimentelor de recunoaştere a vorbitorilor folosind CNN au fost de fiecare dată superioare celor obţinute folosind modelele clasice de reţele neurale. Pe de altă parte, rezultatele SOM-CNN au fost mai bune cu aproximativ 4% faţa de cele ale MLP-CNN şi cu 1% faţă de TDNN-CNN.

Am testat SOM şi SOM-CNN pentru recunoaşterea unor vocale din limba română şi am constatat că procentul vocalelor şi semivocalelor recunoscute de SOM-CNN este superior celui obţinut cu SOM cu mai mult de 10%. Regiunile din semnal corespunzătoare vocalelor şi semivocalelor sunt mai cuprinzătoare atunci când se foloseşte SOM-CNN decât atunci când se foloseşte SOM.

Am văzut în acest capitol că reţelele neurale concurente implementează conceptul de concurenţă la nivelul unei colecţii de reţele neurale. Experimentele demonstrează că modelul reţelelor neurale concurente este superior în majoritatea cazurilor folosirii unei singure reţele neurale pentru rezolvarea aceleiaşi probleme.

În următoarele două capitole vom atinge a doua problemă importantă a tezei şi vom introduce o serie de algoritmi prin care se poate determina importanţa intrărilor pentru reţelele neurale care folosesc algoritmi de instruire tip LVQ.

Capitolul 5

Recunoaştere prin calculul relevanţelor intrărilor folosind operatori OWA

Algoritmul Relevance Learning Vector Quantization (RLVQ) [10] este o variantă a algoritmului LVQ care permite determinarea printr-o metodă euristică a relevanţelor pentru componentele intrărilor. Metoda se bazează pe învăţarea hebbiană şi introduce factorii de ponderare a intrărilor, adaptaţi automat la problema de clasificare care trebuie implementată. Pe lângă stabilirea importanţei intrărilor, această metodă conduce la îmbunătăţirea performanţelor algoritmului LVQ dar, în acelaşi timp, poate fi folosită pentru reducerea numărului de intrări prin renunţarea la cele mai puţin influente.

În acest capitol propunem o nouă metodă de calcul al relevanţelor intrărilor în RLVQ. Relevanţele sunt calculate on-line ca ponderi ale unui operator Ordered Weighted Aggregation (OWA). Operatorii OWA reprezintă o familie de operatori de agregare şi au fost definiţi de Yager [93]. Principalul avantaj al acestui algoritm numit OWA-RLVQ este acela că face legătura dintre RLVQ şi consistentul model matematic OWA.

5.1 Introducere

Kohonen a introdus LVQ ca o metodă care defineşte o clasificare pe baza unui număr de vectori grupaţi într-un set de instruire. Aşa cum am văzut deja, cuantizarea vectorială înseamnă înlocuirea vectorilor de instruire cu un număr redus de prototipuri care formează un *codebook*. Fiecare vector din *codebook* are ataşată eticheta clasei din care face parte. Algoritmul LVQ adaptează iterativ

aceşti vectori prin optimizarea unui criteriu global bazat pe distanţa euclidiană.

Algoritmul LVQ standard nu face distincţia între componentele vectoriale mai influente şi cele mai puţin influente, importanţa lor fiind considerată egală. Spre deosebire de această abordare, algoritmul *Distinction Sensitive Learning Vector Quantizer (DSLVQ)* introdus în [68] foloseşte o pondere modificabilă pentru fiecare componentă vectorială şi foloseşte o funcţie de distanţă ponderată pentru clasificare. Se utilizează un algoritm euristic iterativ pentru adaptarea acestor ponderi la specificul problemei: influenţa componentelor care în mod frecvent generează o clasificare greşită este diminuată, în timp ce influenţa componentelor care contribuie la clasificarea corectă este amplificată. Procesul de ponderare poate fi privit ca o transformare prin scalare a spaţiului vectorilor de instruire într-un spaţiu bazat pe o distanţă ponderată: creşte distanţa dintre valorile componentelor cele mai puternic discriminative şi o micşorează pe cea dintre valorile componentelor care aduc mai puţină informaţie. Aceasta facilitează distincţia dintre clase şi face sistemul mai puţin sensibil la zgomot [69].

Algoritmul DSLVQ foloseşte o funcţie de distanţă ponderată care se bazează pe distanţa euclidiană ponderată. Se pot folosi, însă, şi alte distanţe, după cum se precizează în [68]. Modificarea algoritmului LVQ este minimală, fiind înlocuită funcţia distanţei. Ponderile şi vectorii codebook se modifică în paralel în timpul instruirii, actualizarea având loc on-line, după procesarea fiecărui vector de intrare. Ponderile rezultate pot fi interpretate ca ranguri ale componentelor vectoriale şi folosite pentru reducerea dimensiunii spaţiului de intrare.

RLVQ este o variantă modificată a LVQ, similară cu DSLVQ, şi introduce factorii de relevanţă pentru pentru fiecare dimensiune a vectorilor de intrare. Metrica euclidiană este înlocuită cu una modificată prin asocierea unei ponderi fiecărei componente vectoriale. Adiţional algoritmului standard, factorii de scalare sunt adaptaţi iterativ printr-o tehnică bazată pe învăţarea hebbiană, oferind o imagine a influenţei fiecărei componente vectoriale asupra procesului de clasificare.

Operatorii OWA reprezintă o clasă de operatori de agregare care realizează o agregare bazată pe reordonarea criteriilor care trebuie satisfăcute. Ponderile operatorului OWA sunt asociate poziţiilor acestor criterii obţinute în urma reordonării, şi nu unui criteriu anume.

Noul algoritm de calcul al relevanţelor pe care îl introducem în acest capitol calculează importanţele componentelor de intrare considerându-le ponderi ale unui operator OWA. Ponderile sunt adaptate on-line, în paralel cu instruirea prototipurilor. Metrica pe care o folosim este un caz particular al mediei generalizate de tip OWA [37]. Pe lângă recunoaşterea eficientă a formelor, metoda

noastră face şi o conexiune interesantă între două abordări diferite: RLVQ şi OWA.

Experimentele prezentate în finalul capitolului demonstrează capacitatea tehnicii propuse de noi de a rezolva probleme dificile, iar rezultatele confirmă acest lucru. Am folosit trei baze de date care reprezintă standarde în clasificarea formelor. Cu bazele de date Iris şi Ionosphere am testat performanţele generale în recunoaştere ale algoritmului nostru. Pentru testele de recunoaştere a vorbirii am utilizat baza de date Deterding care conţine un set de vectori de referinţă în domeniu, obţinuţi din pronunţii ale vocalelor în diferite contexte.

5.2 Algoritmul Relevance LVQ (RLVQ)

Uneori nu toate componentele vectorilor de intrare au aceeaşi importanţă în decizia pe care o ia un sistem de clasificare. Unele se pot dovedi mai influente decât altele, astfel încât putem deveni interesaţi în a calcula ponderile lor.

RLVQ este un algoritm euristic, supervizat, care amplifică valoarea ponderilor acelor componente care contribuie cel mai mult la clasificarea corectă a vectorilor de instruire şi slăbeşte ponderile componentelor care au o influenţă negativă. Clustering-ul este realizat de un set de prototipuri care sunt adaptate vectorilor de instruire printr-un algoritm LVQ standard. Figura 5.1 este o reprezentare schematică a modului în care lucrează acest algoritm.

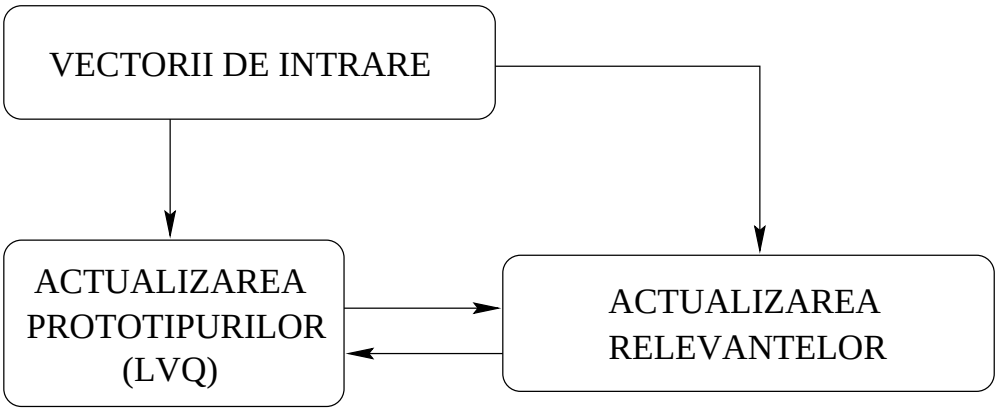

Figura 5.1: Reprezentarea schematică a modului în care se actualizează relevanţele şi prototipurile prin algoritmul RLVQ.

Presupunem că $\mathbf{x}_i = [x_{i1}, \ldots, x_{in}]^t$, $i = 1, \ldots, N$ sunt vectorii de intrare. Un număr de P prototipuri sunt adaptate prin LVQ1 astfel încât să se obţină cea mai mică eroare de cuantizare posibilă pentru toţi vectorii de instruire:

Pasul 1. *Pentru o intrare* $\mathbf{x}_i$ *se găseşte cel mai apropiat prototip* $\mathbf{w}_j$*, câştigătorul, pentru care distanţa* $\|\mathbf{x}_i - \mathbf{w}_j\|$ *este minimă. Dacă* $\mathbf{x}_i$ *şi* $\mathbf{w}_j$ *fac parte din aceeaşi clasă, atunci vectorul de intrare este clasificat corect.*

Pasul 2. *Se actualizează prototipul câştigător:*

$$\mathbf{w}_j = \begin{cases} \mathbf{w}_j + \eta(\mathbf{x}_i - \mathbf{w}_j) & \text{dacă } \mathbf{x}_i \text{ este clasificat corect} \\ \mathbf{w}_j - \eta(\mathbf{x}_i - \mathbf{w}_j) & \text{în caz contrar,} \end{cases}$$

unde $\eta > 0$ *este rata de instruire.*

Algoritmul RLVQ foloseşte o metrică modificată:

$$\|\mathbf{x}_i - \mathbf{w}_j\|_\lambda = \sqrt{\sum_{k=1}^{n} \lambda_k (x_{ik} - w_{jk})^2},$$

unde $\lambda = [\lambda_1, \ldots, \lambda_n]^t$ este vectorul relevanţelor, iar $\sum_{k=1}^{n} \lambda_k = 1$. Urmând o regulă similară algoritmului LVQ, factorii de ponderare sunt adaptaţi iterativ astfel [10]:

Pasul 1. *Pentru fiecare* $k = 1, \ldots, n$

$$\lambda_k = \begin{cases} max\{\lambda_k - \alpha|x_{ik} - w_{jk}|, 0\} & \text{când } \mathbf{x}_i \text{ este clasificat corect} \\ \lambda_k + \alpha|x_{ik} - w_{jk}| & \text{în rest,} \end{cases}$$

unde $\alpha > 0$ *este un parametru de instruire.*

Pasul 2. *Ponderile* λ_k *sunt normalizate pentru fiecare* $k = 1, \ldots, n$.

Determinarea relevanţelor poate fi realizată după instruirea prototipurilor cu LVQ sau simultan cu aceasta, cea de-a doua variantă conducând la un algoritm on-line. Rezultatele experimentale din [10] demonstrează că rata de recunoaştere a RLVQ este mai bună în comparaţie cu algoritmul LVQ standard.

Această regulă de actualizare a relevanţelor foloseşte principiul învăţării hebbiene, aşa cum se demonstrează în [10].

Atunci când vectorul $\mathbf{x}_i$ este clasificat corect, ponderile care corespund diferenţelor mari între componentele lui $\mathbf{x}_i$ şi ale lui $\mathbf{w}_j$ sunt diminuate cu o valoare mai mare. Ponderile corespunzătoare unor diferenţe pe componente mai mici sunt diminuate mai puţin, exploatând astfel similaritatea componentelor vectoriale. Prin aplicarea normalizării, vor fi amplificate relevanţele componentelor care contribuie cel mai mult la clasificarea corectă.

Atunci când clasificarea este incorectă, relevanţele corespunzătoare diferenţelor mici trebuie să piardă din importanţă, crescând cu o valoare mică. Cele

corespunzătoare diferenţelor mari sunt asociate componentelor cu o pondere mai mare în clasificare şi sunt modificate tot în sens crescător, dar cu o valoare mai mare. Prin normalizare, acele relevanţe care contribuie la clasificarea incorectă sunt diminuate cel mai mult.

5.3 Operatorii OWA

În această secţiune vom descrie conceptul de operator OWA [93], proprietăţile fundamentale ale acestuia [93] şi vom relua o metodă propusă de Yager în [21] pentru instruirea operatorilor OWA pe baza unei colecţii de eşantioane.

5.3.1 Problema generală a agregării

Presupunem că $C_1, \ldots, C_n$ sunt n criterii care definesc o problemă multicriterială. Notăm cu X domeniul în care iau valori argumentele celor n criterii şi cu I intervalul $[0, 1]$. Problema constă în formularea unei funcţii globale de decizie D în aşa fel încât pentru orice alternativă $x \in X$, funcţia $D(x) \in I$ indică gradul în care x îndeplineşte condiţiile cerute în raport cu cele n criterii, atunci când $C_i(x) \in I$, $i = 1, \ldots, n$ indică gradul în care x satisface criteriul C_i. Putem scrie, aşadar:

$$D(x) = F(C_1(x), \ldots, C_n(x)),$$

unde funcţia F este un operator de agregare care îndeplineşte condiţiile de:

a) Monotonie. Cu cât un criteriu individual este satisfăcut într-o măsură mai mare, cu atât valoarea funcţiei globale de decizie creşte:

$$\text{dacă } C_i(x) \geqslant C_i(y)\ \forall i = 1, \ldots, n \text{ şi } \ x, y \in X \Rightarrow D(x) \geqslant D(y);$$

b) Simetrie. Ordinea în care sunt tratate criteriile de către funcţia globală de decizie nu este importantă, cu alte cuvinte un argument particular al funcţiei F poate apărea pe orice poziţie în lista de argumente ale lui F.

Analiza unui operator de agregare presupune studiul relaţiei care există între criteriile care descriu problema. O situaţie extremă este cea în care o alternativă x trebuie să satisfacă toate criteriile $C_1, \ldots, C_n$. În acest caz este vorba despre un *anding* aplicat celor n valori. Un exemplu de astfel de operator este $D(x) = Min(C_1(x), \ldots, C_n(x))$. Cealaltă extremă este situaţia în care trebuie satisfăcut cel puţin un criteriu, cerinţă care presupune un *oring* al valorilor, operatorul $D(x) = Max(C_1(x), \ldots, C_n(x))$ făcând parte din această categorie. Majoritatea cazurilor se găsesc între cele două extreme, adică între cerinţa de a îndeplini toate criteriile şi cea de a îndeplini cel puţin unul dintre ele.

5.3.2 Conceptul de operator OWA

O categorie specială de operatori de agregare sunt operatorii OWA [93].

Un operator OWA de dimensiune n este o funcţie

$$F : R^n \to R$$

$$F(a_1, \ldots, a_n) = \sum_{i=1}^{n} w_i a_i^*,$$

unde $\mathbf{W} = [w_1 \ldots w_n]^t$ este un vector de ponderi asociat operatorului, cu $w_i \in [0, 1]$, $\sum_{i=1}^{n} w_i = 1$, iar $\mathbf{A}^* = [a_1^* \ldots a_n^*]^t$ este o reordonare descrescătoare a argumentelor lui $\mathbf{A} = [a_1 \ldots a_n]^t$ în aşa fel încât a_i^* este al i-lea cel mai mare dintre a_i, $a_1^* \geqslant \ldots \geqslant a_n^*$.

Elementul central în utilizarea unui operator OWA este pasul de reordonare a argumentelor în urma căruia un argument particular a_i nu mai este asociat unei ponderi particulare w_i, ci ponderea w_i este asociată valorii de pe poziţia i obţinută în urma reordonării.

Se demonstrează [93] că operatorii OWA satisfac proprietăţile de comutativitate, monotonie şi idempotenţă. O proprietate importantă a acestor operatori este că ei includ operatorii Min, Max şi media aritmetică pentru anumite valori particulare ale matricii $\mathbf{W}$:

a) Pentru $\mathbf{W} = [0 \ldots 0\ 1]^t$, $F(a_1, \ldots, a_n) = Min(a_1, \ldots, a_n)$;

b) Pentru $\mathbf{W} = [1\ 0 \ldots 0]^t$, $F(a_1, \ldots a_n) = Max(a_1, \ldots, a_n)$;

c) Pentru $\mathbf{W} = [1/n \ldots 1/n]^t$, $F(a_i, \ldots, a_n) = \frac{1}{n} \sum_{i=1}^{n} a_i$.

Operatorii OWA au, de asemenea, proprietatea de mărginire:

$$Min(a_1, \ldots, a_n) \leqslant F(a_1, \ldots, a_n) \leqslant Max(a_1, \ldots, a_n).$$

Pornind de la această inegalitate, a fost introdusă în [93] o măsură care caracterizează tipul de agregare realizat de o anumită valoare a vectorului de ponderi $\mathbf{W}$. Această măsură se numeşte *orness* şi se defineşte prin relaţia:

$$O(\mathbf{W}) = \frac{1}{n-1} \sum_{i=1}^{n} (n - i)\ w_i.$$

Domeniul său de valori este $[0, 1]$ şi se apropie de 1 cu atât mai mult cu cât operatorul OWA are caracter mai pronunţat al unui *or*, adică al funcţiei Max. Pentru setul de valori ale lui $\mathbf{W}$ care corespund operatorilor Min, Max şi media aritmetică, valorile lui $O(\mathbf{W})$ sunt:

a) $O([0 \ldots 0\ 1]^t) = 0$;

b) $O([1\ 0 \ldots 0]^t) = 1$;

c) $O([1/n \ldots 1/n]^t) = 0{,}5$.

O altă măsură introdusă în [93] este *dispersia*:

$$D(\mathbf{W}) = \sum_{i=1}^{n} w_i \ln w_i$$

care a fost propusă pentru calculul cantităţii de informaţie înglobată de ponderile $\mathbf{W}$ ale unui operator OWA.

5.3.3 Instruirea operatorilor OWA

Au fost propuse diverse metode de alegere a ponderilor operatorilor OWA. O'Hagan [21] a introdus o tehnică bazată pe un *orness* prestabilit şi o maximizare a dispersiei. Torra [85], [86] a utilizat un operator de agregare particular numit Weighted OWA şi i-a determinat parametrii printr-o procedură care generează câte o ieşire ideală pentru fiecare vector de instruire. Karayiannis [37] a folosit două familii de ponderi OWA care cuprind, prima, un set de ponderi egale completat cu un număr de ponderi cu valoarea 0, iar a doua, ponderi care descresc liniar completate, de asemenea, cu un număr de valori de 0. Beliakov [6] a aproximat operatorii OWA rezolvând o problemă de tip Least Squares with Equality and Inequality (LSEI).

Filev şi Yager [21] au calculat ponderile operatorilor OWA prin metoda gradientului descrescător folosind o valoare ţintă pentru valoarea agregată. Prezentăm pe scurt această metodă.

Algoritmul presupune existenţa a m eşantioane alcătuite din valori ale celor n criterii, $[a_{i1} \ldots a_{in}]^t$, $i = 1, \ldots, m$ şi a unei valori d_i asociate fiecărui eşantion, numită *valoare agregată*. Metoda prin care se fixează valoarea agregată nu are relevanţă pentru acest algoritm. De exemplu, ea poate fi apreciată de un expert care, studiind scorul acordat fiecărui criteriu, stabileşte o valoare globală.

Scopul algoritmului este de a obţine ponderile unui operator OWA care modelează procesul de agregare ce a generat setul de date de instruire.

Presupunem că $[a^*_{i1} \ldots a^*_{in}]^t$ reprezintă o reordonare descrescătoare a criteriilor $[a_{i1} \ldots a_{in}]^t$. Atunci valoarea agregată va fi:

$$d_i = a^*_{i1} w_1 + \ldots + a^*_{in} w_n, \ \forall\, i = 1, \ldots, m.$$

Rescriem această relaţie în sensul unei erori pătratice medii pe care vom căuta să o minimizăm:

$$e_i = \frac{1}{2} \left(a^*_{i1} w_1 + \ldots + a^*_{in} w_n - d_i \right)^2,$$

cu $w_j \in [0,1]$, $j = 1, \ldots, n$ şi $\sum_{j=1}^{n} w_j = 1$. Pentru a elimina aceste două constrângeri care definesc un operator OWA, presupunem că ponderile sunt definite astfel:

$$w_j = \frac{e^{\lambda_j}}{\sum_{k=1}^{n} e^{\lambda_k}},\ j = 1, \ldots, n.$$

Astfel, problema de minimizare cu constrângeri enunţată mai sus se rescrie în felul următor:

$$e_i = \frac{1}{2}\left(a_{i1}^{*}\frac{e^{\lambda_1}}{\sum_{k=1}^{n} e^{\lambda_k}} + \ldots + a_{in}^{*}\frac{e^{\lambda_n}}{\sum_{k=1}^{n} e^{\lambda_k}} - d_i\right)^2. \tag{5.1}$$

Eroarea pătratică medie descrisă de relaţia (5.1) trebuie minimizată în funcţie de parametrii $\lambda_1, \ldots, \lambda_n$.

Pentru rezolvarea acestei probleme de optimizare se foloseşte metoda gradientului descrescător şi se obţine regula de actualizare a parametrilor λ_j, $j = 1, \ldots, n$:

$$\lambda_j^{(t+1)} = \lambda_j^{(t)} - \eta\frac{\partial e_i}{\partial \lambda_j},$$

unde t este pasul de instruire şi η este rata de instruire. Notând cu $\hat{d}_i$ valoarea estimată a lui d_i:

$$\hat{d}_i = a_{i1}^{*}\frac{e^{\lambda_1}}{\sum_{k=1}^{n} e^{\lambda_k}} + \ldots + a_{in}^{*}\frac{e^{\lambda_n}}{\sum_{k=1}^{n} e^{\lambda_k}}$$

rezultă:

$$\frac{\partial e_i}{\partial \lambda_j} = w_j\left(a_{ij} - \hat{d}_i\right)\left(\hat{d}_i - d_i\right),\ i = 1, \ldots, n$$

sau

$$\lambda_j^{(t+1)} = \lambda_j^{(t)} - \eta w_j\left(a_{ij} - \hat{d}_i\right)\left(\hat{d}_i - d_i\right)$$

cu

$$w_j = \frac{e^{\lambda_j^{(t)}}}{\sum_{k=1}^{n} e^{\lambda_k^{(t)}}},\ j = 1, \ldots, n.$$

Parametrii λ_j, $j = 1, \ldots, n$ sunt actualizaţi prin propagarea erorii $\hat{d}_i - d_i$ dintre valoare agregată $\hat{d}_i$ estimată de ponderile operatorului OWA obţinute până la momentul t şi valoarea agregată reală d_i.

Această metodă calculează ponderile unui operator OWA având la dispoziţie un set de eşantioane care constau din valori ale celor n criterii şi câte o valoare agregată pentru fiecare eşantion.

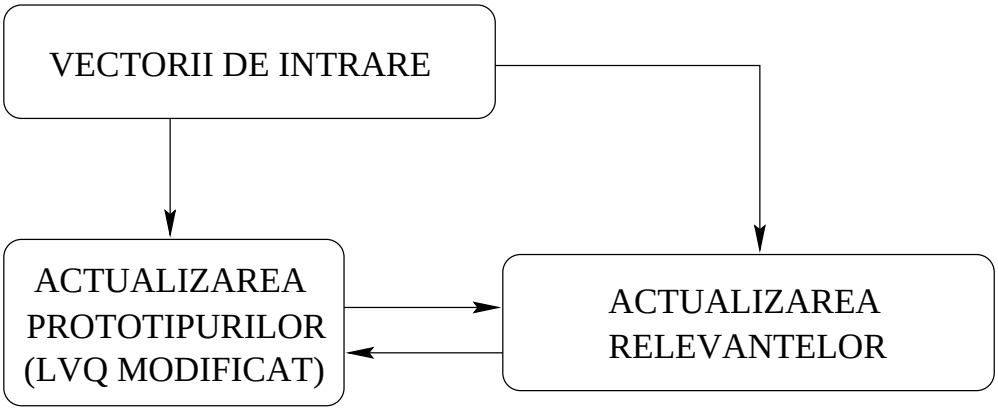

Figura 5.2: Reprezentarea schematică a modului în care se actualizează relevanţele şi prototipurile prin algoritmul OWA-RLVQ. Algoritmul LVQ este modificat prin înlocuirea distanţei euclidiene cu distanţa D^*_{ij}, iar relevanţele sunt calculate ca ponderi ale unui operator OWA.

5.4 Algoritmul OWA - Relevance LVQ (OWA-RLVQ)

În această secţiune introducem un nou algoritm [14] care calculează ponderile unui operator OWA ca relevanţe ale componentelor vectorilor de instruire pentru probleme de clustering supervizat, aşa cum se poate vedea în figura 5.2.

Fie $\mathbf{x}_i \in \mathbf{R}^n$ un vector de instruire, $i = 1, \ldots, N$, N fiind numărul de vectori de instruire. Considerăm $\mathbf{w}_k$, $k = 1, \ldots, P$ prototipurile care pot fi calculate printr-un algoritm LVQ. Pentru a calcula ponderile OWA, vom considera ca operator de agregare următoarea distanţă modificată:

$$D^*_{ij} = \sqrt{\sum_{k=1}^{n} \lambda_k (|x_{ik} - w_{jk}|^*)^2},$$

unde $\sum_{k=1}^{n} \lambda_k = 1$ şi $|x_{ik} - w_{jk}|^*$ este cea de-a k-a cea mai mare distanţă dintre componentele corespondente ale vectorului de instruire şi vectorului prototip.

Această distanţă este un caz special al mediei generalizate de tip OWA [37]:

$$M = \left(\sum_{k=1}^{n} \lambda_k a_k^{*p} \right)^{\frac{1}{p}}, \tag{5.2}$$

unde $p \in \mathbf{R}^*$. Pentru $p = 2$, (5.2) se reduce la distanţa modificată D^*_{ij} propusă de noi dacă facem următoarele înlocuiri:

$$a^*_k = |x_{ik} - w_{jk}|^*$$

şi

$$|x_{i1} - w_{j1}|^* \geqslant \ldots \geqslant |x_{in} - w_{jn}|^*.$$

Algoritmul LVQ1 trebuie reformulat pentru a minimiza o funcţie obiectiv bazată pe această distanţă modificată. Calculăm:

$$\Delta \mathbf{w}_j^* = \eta \lambda \mathbf{I}(\mathbf{x}_i - \mathbf{w}_j)^*$$

dacă $\mathbf{x}_i$ este clasificat corect conform distanţei D_{ij}^* şi:

$$\Delta \mathbf{w}_j^* = -\eta \lambda \mathbf{I}(\mathbf{x}_i - \mathbf{w}_j)^*$$

dacă $\mathbf{x}_i$ nu este clasificat corect. În relaţiile de mai sus am notat cu $(\mathbf{x}_i - \mathbf{w}_j)^*$ vectorul diferenţă ale cărui componente sunt reordonate, iar relevanţele se vor aplica după reordonare. Din această cauză, setul de prototipuri obţinut în urma acestei proceduri nu va fi acelaşi cu cel obţinut în urma algoritmului LVQ standard pentru că reflectă două abordări diferite ale instruirii.

Un vector prototip $\mathbf{w}_j$ va eticheta cu numele clasei sale toţi vectorii de intrare care îndeplinesc următoarea inegalitate:

$$|\mathbf{x}_i - \mathbf{w}_j|^* < |\mathbf{x}_i - \mathbf{w}_l|^*, \forall j \neq l.$$

Fie

$$\lambda = [\lambda_1 \dots \lambda_n]^t$$

$$\mathbf{x}_i = [x_{i1} \dots x_{in}]^t$$

$$\mathbf{w}_j = [w_{j1} \dots w_{jn}]^t$$

şi

$$\mathbf{d} = [d_1 \dots d_n]^t,$$

unde $d_k = x_{ik} - w_{jk}$, $k = 1, \dots, n$. Dacă $\mathbf{x}_i$ este clasificat corect conform distanţei D_{ij}^*, o valoare mică a lui $|d_k|^*$ ar trebui să inducă o valoare mare pentru $\Delta\lambda_k$, iar o valoare mare a lui $|d_k|^*$ ar trebui să conducă la o valoare mică pentru $\Delta\lambda_k$. Astfel, pentru o clasificare corectă conform criteriului formulat anterior, $|d_k|^* \leqslant |d_{k'}|^*$ implică $\Delta\lambda_k \geqslant \Delta\lambda_{k'}$, adică $-|d_k|^* \geqslant -|d_{k'}|^*$ înseamnă $\Delta\lambda_k \geqslant \Delta\lambda_{k'}$ pentru $k, k' = 1, \dots, n$. Putem, astfel, să luăm:

$$\Delta\lambda_k = -\alpha |d_k|^*$$

sau:

$$\Delta\lambda_k = -\alpha |x_{ik} - w_{jk}|^*,$$

unde $\alpha > 0$ este un parametru de instruire. Atunci când clasificarea conform aceluiaşi criteriu este incorectă, $|d_k|^* \leqslant |d_{k'}|^*$ implică $\Delta\lambda_k \leqslant \Delta\lambda_{k'}$, pentru $k, k' = 1, \dots, n$. Putem lua:

$$\Delta\lambda_k = \alpha |d_k|^*$$

sau:

$$\Delta\lambda_k = \alpha|x_{ik} - w_{jk}|^*.$$

Folosim transformarea

$$\lambda_k = \frac{e^{\lambda_k}}{\sum_{i=1}^{n} e^{\lambda_i}}$$

pentru $i = 1, \ldots, n$ pentru a ne asigura că $\sum_{k=1}^{n} \lambda_k = 1$ şi $\lambda_k \in [0, 1]$ care sunt constrângerile impuse ponderilor operatorilor OWA.

Următoarea procedură actualizează on-line atât relevanţele cât şi rangul fiecărei componente a vectorilor de intrare.

Pasul 1. *Iniţializează η şi α. Iniţializează $\lambda_k = \frac{1}{n}$, $k = 1, \ldots, n$, unde n este numărul de componente ale vectorului de intrare.*

Pasul 2. *Iniţializează prototipurile.*

Pasul 3. *Actualizează prototipurile conform regulii LVQ modificate:*

$$\mathbf{w}_j^* = \begin{cases} \mathbf{w}_j^* + \eta\lambda\mathbf{I}(\mathbf{x}_i - \mathbf{w}_j)^* & \text{dacă } \mathbf{x}_i \text{ este clasificat corect} \\ & \text{conform distanţei } D_{ij}^* \\ \mathbf{w}_j^* - \eta\lambda\mathbf{I}(\mathbf{x}_i - \mathbf{w}_j)^* & \text{în rest.} \end{cases}$$

Pasul 4. *Actualizează relevanţele:*

$$\lambda_k = \begin{cases} \lambda_k - \alpha|x_{ik} - w_{jk}|^* & \text{dacă } \mathbf{x}_i \text{ este clasificat corect} \\ & \text{conform distanţei } D_{ij}^* \\ \lambda_k + \alpha|x_{ik} - w_{jk}|^* & \text{în rest} \end{cases}$$

pentru $k = 1, \ldots, n$.

Pasul 5. *Transformă relevanţele:*

$$\lambda_k = \frac{e^{\lambda_k}}{\sum_{i=1}^{n} e^{\lambda_i}}$$

pentru $k = 1, \ldots, n$.

Pasul 6. *Calculează ponderea fiecărei componente a vectorului de intrare ca medie a indicilor de poziţie calculaţi înaintea pasului de ordonare, pentru toţi paşii parcurşi de la începutul instruirii.*

Pasul 7. *Repetă paşii 3–6 pentru fiecare vector de instruire.*

Acest algoritm calculează ponderile unui operator OWA ca relevanţe prin minimizarea distanţei modificate D_{ij}^* care este şi valoarea agregată. Trebuie să facem distincţia între relevanţe şi rangul unei componente. Rangul este ataşat unei componente. Relevanţa este ataşată unei poziţii specifice din vectorul ordonat al distanţei pe componente dintre un vector de intrare şi un prototip. Aceasta explică semnificaţia pasului 6.

5.5 Experimente

În studiul nostru am testat algoritmul OWA-RLVQ pe baze de date standard pentru a vedea care este structura tipică a vectorilor de relevanţe şi care sunt performanţele în recunoaştere. Programul care implementează acest algoritm a fost scris în limbajul C++. Instruirea este on-line, fiecare vector fiind prezentat o singură dată, astfel încât timpul de execuţie este practic neglijabil.

Baza de date Iris [9] conţine 3 clase alcătuite din câte 50 de vectori. Două dintre ele nu sunt liniar separabile. Problema este detectarea claselor pe baza celor 4 componente vectoriale. Am instruit OWA-RLVQ cu 6 prototipuri şi am obţinut o rată medie de recunoaştere de 96,6%. Un vector tipic al relevanţelor este $[0{,}15\ 0{,}21\ 0{,}23\ 0{,}38]^t$. Am luat η=0,3 şi α=0,2. Experimentele au arătat că a patra componentă vectorială are influenţa cea mai mare, iar prima componentă vectorială este cea mai puţin semnificativă, aceste rezultate fiind confirmate şi de cele din [10]. Rangurile obţinute pentru fiecare componentă după instruirea cu RLVQ şi OWA-RLVQ folosind acelaşi set de prototipuri iniţiale sunt prezentate în tabelul 5.1.

Tabelul 5.1: Rangul componentelor vectoriale pentru baza de date Iris.

Rangul	1	2	3	4
Componentele RLVQ	4	2	3	1
Componentele OWA-RLVQ	4	3	2	1
Ponderile componentelor OWA-RLVQ	1,86	1,44	1,24	1,17

Baza de date Deterding (Vowel Recognition) [9] conţine vectori extraşi din pronunţiile vocalelor în 11 contexte. Fiecare pronunţie este repetată de 6 ori de cei 15 vorbitori. Problema cere să se folosească pronunţiile primilor 8 vorbitori pentru instruire şi pronunţiile ultimilor 7 vorbitori pentru recunoaştere. Un vector tipic al relevanţelor obţinut prin algoritmul OWA-RLVQ este $[0{,}032\ 0{,}039\ 0{,}043\ 0{,}044\ 0{,}077\ 0{,}136\ 0{,}137\ 0{,}15\ 0{,}163\ 0{,}173]^t$. Rata de recunoaştere pe care am obţinut-o cu acest algoritm a fost de 46,75%. Rata medie de recunoaştere pentru LVQ1 la acest experiment a fost de 44,8%, iar pentru RLVQ a fost de 46,32%. Problema este cunoscută ca fiind una dificilă, de aceea am folosit 59 de prototipuri. În experimentele OWA-RLVQ am luat η=1,7 şi α=1,9. Rangurile obţinute pentru această bază de date sunt prezentate în tabelul 5.2. Componenta 2 este considerată cea mai importantă de RLVQ şi pe poziţia a doua de OWA-RLVQ, după cum se poate remarca studiind aceste rezultate.

Componenta 10 este cea mai puţin importantă pentru ambele modele.

Tabelul 5.2: Rangul componentelor vectoriale pentru baza de date Deterding.

Rangul	1	2	3	4	5
Componentele RLVQ	2	5	1	9	6
Componentele OWA-RLVQ	8	2	4	5	6
Ponderile componentelor OWA-RLVQ	5,209	5,209	5,20	5,17	5,16
Rangul	6	7	8	9	10
Componentele RLVQ	3	4	8	7	10
Componentele OWA-RLVQ	9	3	7	1	10
Ponderile componentelor OWA-RLVQ	5,15	5,15	5,14	5,14	5,13

Baza de date Ionosphere [9] constă din 351 de vectori de date cu câte 34 de caracteristici. Vectorilor le sunt asociate etichetele "bad" şi "good", fiind vorba de o clasificare binară. Problema constă în folosirea primilor 200 de vectori, echilibraţi numeric între exemple pozitive şi negative, pentru instruire şi ultimii 151 pentru test. Am folosit 8 prototipuri, iar valorile parametrilor pentru instruirea OWA-RLVQ au fost η=3,3 şi α=3,5. Rata medie de recunoaştere a fost de 93,37%, în timp ce pentru LVQ1 am obţinut 90,06%, iar pentru RLVQ am obţinut 92,71%. Ratele uzuale de recunoaştere corectă pentru această bază de date încep, aşa cum se arată în [9], de la 90%. În acest context, rezultatele obţinute de noi sunt foarte bune. Rangurile celor mai importante 5 componente vectoriale obţinute cu RLVQ şi OWA-RLVQ sunt prezentate în tabelul 5.3.

Tabelul 5.3: Rangul componentelor vectoriale pentru baza de date Ionosphere. Sunt reprezentate cele mai importante 5 componente.

Rangul	1	2	3	4	5
Componentele RLVQ	20	28	26	12	6
Componentele OWA-RLVQ	14	12	1	3	28
Ponderile componentelor OWA-RLVQ	19,20	19,16	19,10	19,07	19,06

Tabelul 5.4 compară ratele de recunoaştere obţinute cu LVQ1, RLVQ şi OWA-RLVQ folosind de fiecare dată acelaşi set iniţial de prototipuri. În general am obţinut rezultate mai bune cu algoritmul OWA-RLVQ faţă de cele obţinute

cu RLVQ. Ambele metode sunt superioare algoritmului LVQ1, fapt care ilustrează utilitatea relevanţelor în clasificare. Trebuie accentuată observaţia că valorile relevanţelor obţinute cu RLVQ şi OWA-RLVQ nu se suprapun în mod obligatoriu pentru că reflectă două abordări diferite. În cazul metodei propuse de noi, relevanţele se referă la distanţele reordonate, în timp ce algoritmul RLVQ generează un set de valori asociate în mod precis anumitor componente ale vectorilor de intrare.

Tabelul 5.4: Comparaţie între ratele de recunoaştere obţinute cu LVQ1, RLVQ şi OWA-RLVQ.

Baza de date	LVQ1	RLVQ	OWA-RLVQ
Iris	91,33%	95,33%	96,60%
Deterding	44,80%	46,32%	46,75%
Ionosphere	90,06%	92,71%	93,37%

S-a constatat că, uneori, componentele care au valori neschimbate în tot setul de vectori de instruire pot conduce la falsa accentuare a relevanţei lor, această problemă fiind cunoscută şi pentru RLVQ. Din acest motiv, se poate recurge la o preprocesare în scopul eliminării lor.

5.6 Concluzii

În acest capitol am introdus o nouă metodă de calcul al relevanţelor dimensiunilor vectorilor de intrare ca ponderi OWA. Algoritmul propus calculează on-line relevanţele, oferind posibilitatea unei permanente adaptări la datele de intrare. Rezultatele experimentale au arătat că algoritmul propus de noi oferă o bună rată de recunoaştere. În acelaşi timp, factorii de relevanţă pot fi utilizaţi pentru stabilirea on-line a rangului fiecărei dimensiuni a vectorilor de intrare.

Algoritmul OWA-RLVQ introdus de noi este diferit de algoritmul LVQ bazat pe operatorii OWA introdus de Karayiannis [37] deoarece noi modificăm metrica LVQ pentru ca să urmărească strategia RLVQ.

Este interesant de observat că abordarea noastră este o nouă metodă de calcul al ponderilor OWA. În comparaţie cu metoda descrisă în [21] şi [94] care foloseşte tehnica gradientului descrescător, noi realizăm acest lucru în mod dinamic.

În capitolul următor vom dezvolta două noi metode care calculează factorii de relevanţă prin maximizarea unor mărimi bazate pe informaţia mutuală şi pe energia informaţională.

Capitolul 6

Recunoaştere prin calculul relevanţelor intrărilor folosind noţiuni de teoria informaţiei

Selecţia componentelor vectorilor de intrare într-un sistem de clasificare este o etapă de preprocesare absolut necesară, în special atunci când avem de a face cu cantităţi foarte mari de date. Introducem în acest capitol algoritmul Energy Relevance LVQ (ERLVQ) bazat pe energia informaţională Onicescu [65] ca o metodă incrementală de instruire pentru clasificare supervizată şi calcul al relevanţelor. Algoritmul Mutual Information Relevance LVQ (MIRLVQ) propune o tehnică alternativă de dezvoltare a acestei metode pornind de la informaţia mutuală.

6.1 Introducere

Informaţia mutuală este o mărime care a fost utilizată pentru a implementa diverse metode de selecţie a intrărilor [5], [49]. În acest caz, informaţia mutuală evaluează "conţinutul informaţional" al fiecărei componente în mod individual, raportat la clasa de ieşire. Metoda de selecţie a componentelor constă în căutarea unui subset de componente relevante din setul initial de componente disponibile. Acest subset se alege în aşa fel încât să maximizeze informaţia mutuală. O importanţă însemnată în acest algoritm o are estimarea informaţiei mutuale. Estimarea informaţiei mutuale este o problemă dificilă datorită funcţiilor de densitate de probabilitate condiţionată şi a complexităţii

computaţionale ridicate. Mulţi algoritmi de selecţie a componentelor vectoriale care se bazează pe informaţia mutuală folosesc histograma ca estimator al densităţii de probabilitate [5], [49] . În spaţii cu multe dimensiuni, însă, histograma nu este practică şi nici foarte precisă [32].

Informaţia mutuală bazată pe entropia Renyi pătratică [70] a fost utilizată recent într-o metodă eficientă de selecţie a caracteristicilor [32]. Acest algoritm numit *toward-optimal feature selection methodology - mutual information (OFS-MI)* constă din două criterii bazate pe informaţia mutuală şi un algoritm de căutare. Primul criteriu numit *feature relevancy criterion (FRC)* are scopul de a selecta componentele relevante, iar al doilea, *feature similarity criterion (FSC)*, este folosit pentru a reduce redundanţa între componentele selectate. Considerăm mulţimea F de componente candidate şi S componentele selectate. Pentru a determina cea mai semnificativă componentă din F, toate componentele candidate sunt ordonate folosind relaţia

$$FRC(f) = I(f, S; C),$$

unde C este mulţimea claselor, f este una dintre componentele din F, iar I este informaţia mutuală. Adăugarea unei componente de intrare cu un aport scăzut în creşterea ratei de clasificare conduce la scăderea FRC. Al doilea criteriu pentru care

$$FSC(f; S) = \text{argmax}\left(\frac{I(f; f_i)}{H(f_i)}\right)$$

măsoară similaritatea dintre componenta f şi submulţimea S, unde $f \notin S$ iar H este entropia. Atunci când $FSC(f; S)$ depăşeşte un prag, se consideră că f este similar cu S. FSC este estimat în acest caz folosind informaţia mutuală pătratică [70] dintre două variabile aleatoare continue.

Torkkola [80], [81], [82], [83], [84] maximizează informaţia mutuală pătratică obţinând o metodă de instruire a unui sistem pentru extragerea caracteristicilor. Un astfel de sistem transformă vectorii de intrare în vectori cu un număr mai mic de dimensiuni prin recalcularea componentelor, fără a afecta performanţele clasificatorului. Diferenţa dintre un sistem de selecţie a caracteristicilor şi unul de extragere (transformare, construcţie, generare) a caracteristicilor este aceea că primul foloseşte mai departe un subset al componentelor vectoriale originale, în timp ce al doilea construieşte vectori cu un număr redus de dimensiuni prin recalcularea acestora, fără a păstra în mod necesar valorile iniţiale. Metoda lui Torkkola adaptează informaţia mutuală pătratică din [70] pentru variabile aleatoare discrete şi o maximizează printr-o metodă de gradient ascendent pentru a obţine o transformare convenabilă.

Algoritmul *Mutual Information Relevance LVQ (MIRLVQ)* propus de noi în acest capitol maximizează informaţia mutuală pătratică pentru a obţine relevanţele care conduc la îmbunătăţirea performanţelor unui clasificator bazat pe un algoritmul LVQ. Algoritmul *Energy Relevance LVQ (ERLVQ)* este similar lui MIRLVQ, însă maximizează o mărime bazată pe energia informaţională. Experimental vom arăta că aceşti algoritmi conduc la rate de recunoaştere mai bune decât algoritmul OWA-RLVQ introdus de noi în capitolul 5.

6.2 Entropia Shannon

În această secţiune vom prezenta câteva noţiuni fundamentale de teoria informaţiei [54] care sunt folosite în dezvoltarea algoritmilor MIRLVQ şi ERLVQ.

Presupunem că X este o variabilă aleatoare, iar x o realizare a sa care poate lua valori din mulţimea

$$\{x_1, \ldots, x_n\}.$$

Fiecare valoare poate să apară cu probabilitatea

$$\{p_1, \ldots, p_n\}$$

unde

$$P(x = x_i) = p_i,\ p_i \geqslant 0,\ i = 1, \ldots, n$$

şi

$$\sum_{x_i \in \{x_1, \ldots, x_n\}} P(x = x_i) = 1.$$

Pentru a simplifica notaţia, îl vom înlocui pe $P(x = x_i)$ cu $P(x_i)$ sau cu $P(x)$.

Dacă X şi Y sunt două variabile aleatoare, realizările fiind perechi ordonate (x, y) de valori cu $x \in \{x_1, \ldots, x_n\}$ şi $y \in \{y_1, \ldots, y_m\}$, atunci $P(x = x_i, y = y_j)$ este probabilitatea de a se realiza evenimentul pentru care x şi y au valorile particulare x_i şi y_j, $i \in \{1, \ldots, n\}$ şi $j \in \{1, \ldots, m\}$. Pentru simplificarea notaţiei, scriem $P(x, y)$ în loc de $P(x = x_i, y = y_j)$. Este cunoscută relaţia:

$$P(x = x_i) = \sum_{y \in \{y_1, \ldots, y_m\}} P(x = x_i, y)$$

sau, mai simplu:

$$P(x) = \sum_{y \in \{y_1, \ldots, y_m\}} P(x, y). \tag{6.1}$$

Probabilitatea ca x să fie egal cu x_i atunci când y este egal cu y_j se numeşte probabilitatea condiţionată şi se notează cu

$$P(x = x_i | y = y_j)$$

unde

$$x_i \in \{x_1, \ldots, x_n\}$$

şi

$$y_j \in \{y_1, \ldots, y_m\}.$$

Probabilitatea condiţionată respectă următoarea egalitate:

$$P(x = x_i | y = y_j) = \frac{P(x = x_i, y = y_j)}{P(y = y_j)}, \text{ unde } P(y = y_j) \neq 0. \tag{6.2}$$

Atunci când $P(y = y_j) = 0$, probabilitatea condiţionată nu este definită. Folosind scrierea simplificată, relaţia (6.2) este echivalentă cu:

$$P(x|y) = \frac{P(x, y)}{P(y)}. \tag{6.3}$$

Spunem că variabilele aleatoare X şi Y sunt independente dacă şi numai dacă:

$$P(x, y) = P(x)P(y).$$

Relaţiile de mai sus pot fi rescrise în diverse forme şi obţinem formule utile în aplicaţiile care folosesc probabilităţi. Pornind de la expresia (6.3), deducem următoarele egalităţi:

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x),$$

din care formulăm teorema lui Bayes:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)},$$

iar din (6.1) şi (6.3) obţinem:

$$P(x) = \sum_{y \in \{y_1, \ldots, y_m\}} P(x, y) = \sum_{y \in \{y_1, \ldots, y_m\}} P(x|y)P(y).$$

Shannon [78] a definit entropia ca o măsură a cantităţii de informaţie produsă de o sursă de informaţie reprezentată printr-un proces markovian.

Presupunem că avem o mulţime de evenimente posibile cu probabilităţile $p_1, \ldots, p_n$. Aceste probabilităţi sunt singurele elemente pe care le cunoaştem. Măsura $H(p_1, \ldots, p_n)$ arată care este incertitudinea în legătură cu ieşirile generate de sursa de informaţie şi îndeplineşte următoarele cerinţe [78]:

a) H este continuă în p_i;

b) Dacă toate valorile lui p_i, $i = 1, \ldots, n$ sunt egale, $p_i = \frac{1}{n}$, atunci incertitudinea este maximă;

c) Dacă "alegerea" unui eveniment de către sursa de informaţie poate fi împărţit în două "alegeri" succesive, atunci valoarea originală a lui H este suma ponderată a valorilor individuale ale lui H.

Shannon a demonstrat că singura funcţie H care satisface cerinţele de mai sus este :

$$H(X) = -k \sum_{x \in \{x_1, \ldots, x_n\}} P(x) \log P(x)$$

sau

$$H(X) = k \sum_{x \in \{x_1, \ldots, x_n\}} P(x) \log \frac{1}{P(x)},$$

cu posibilitatea de a alege $k = 1$ şi cu convenţia că pentru $P(x) = 0$ avem:

$$0 \times \log \frac{1}{0} = 0$$

deoarece

$$\lim_{\theta \to 0^+} \theta \log \frac{1}{\theta} = 0.$$

Funcţia H se numeşte entropie.

Conţinutul informaţional al realizării x a variabilei aleatoare X se defineşte prin:

$$h(x) = -\log_2 P(x).$$

Atât conţinutul informaţional cât şi entropia sunt măsurate în biţi.

Atunci când avem două variabile aleatoare X şi Y, entropia este:

$$H(X,Y) = \sum_{(x,y) \in \{x_1, \ldots, x_n\} \times \{y_1, \ldots, y_m\}} -P(x,y) \log P(x,y).$$

Entropia este aditivă pentru variabile aleatoare independente:

$$H(X,Y) = H(X) + H(Y) \Leftrightarrow P(x,y) = P(x)P(y).$$

Dacă variabilele aleatoare sunt continue, atunci nu mai putem discuta de probabilitatea P a unei anumite realizări şi folosim funcţia de densitate de probabilitate pe care o notăm în continuare cu p.

6.3 Estimarea densităţilor de probabilitate cu ferestre Parzen

În această secţiune vom aborda problema modelării unei funcţii de densitate de probabilitate [8], [17] pe baza unui număr de eşantioane $\mathbf{y}_i$, $i = 1, \ldots, N$ care sunt realizări ale unei variabile aleatoare Y. Vom vedea cum se poate folosi acest

rezultat pentru estimarea entropiei Renyi pătratice ca prim pas în estimarea din eşantioane a informaţiei mutuale dintre două variabile aleatoare.

Presupunem că fiecare eşantion $\mathbf{y}_i$ aparţine uneia dintre clasele $c_1, \ldots, c_P$ care sunt realizări ale variabilei aleatoare discrete C. Spunem că vectorii $\mathbf{y}_i$ sunt etichetaţi cu numele clasei din care fac parte.

Există două importante categorii de metode pentru estimarea densităţii de probabilitate: *metodele parametrice* şi *metodele neparametrice*.

Metodele parametrice pornesc de la presupunerea că funcţia densităţii de probabilitate are o formă cunoscută. Se foloseşte un set de parametri care sunt apoi optimizaţi prin adaptarea modelului la setul de date disponibile. Dezavantajul acestei metode este că forma particulară aleasă pentru funcţia densităţii de probabilitate poate să nu modeleze foarte fidel adevărata reprezentare.

Metodele neparametrice de estimare nu fac vreo presupunere asupra modelului densităţii de probabilitate şi permit ca forma acesteia să fie determinată exclusiv pe baza setului de date de instruire.

Ne vom concentra asupra metodelor neparametrice deoarece dorim să construim un model de recunoaştere care se bazează exclusiv pe fluxul de date, fără a face nici o presupunere asupra proprietăţilor acestor date. Acest gen de aplicaţii sunt caracteristice sistemelor care prelucrează cantităţi mari de date în timp real, fiind capabile totodată să îşi adapteze dinamic parametrii la proprietăţile în schimbare ale intrărilor. Este vorba aşadar, de sisteme care sunt instruite on-line, pentru aplicaţii de data mining unde datele sosesc secvenţial, nefiind stocate din cauza volumului foarte mare şi a costurilor pe care le implică aceasta. Este important de spus că algoritmii on-line oferă posibilitatea de a se renunţa la datele care au fost folosite, deoarece nu se mai revine asupra lor.

Estimarea densităţii de probabilitate prin eşantioane

Reluăm în această secţiune o metodă de estimare descrisă în [8] şi [17].

Fie un vector oarecare $\mathbf{y}$ care face parte din setul de date de instruire şi care contribuie la definirea densităţii de probabilitate $p(\mathbf{y})$. Probabilitatea ca vectorul $\mathbf{y}$ să se găsească în subspaţiul S se defineşte prin:

$$P = \int_S p(\mathbf{y}')d\mathbf{y}'.$$

Dacă presupunem că $p(\mathbf{y})$ este continuă şi nu are variaţii foarte mari pe domeniul S, putem aproxima această probabilitate prin:

$$P = \int_S p(\mathbf{y}')d\mathbf{y}' \simeq p(\mathbf{y})V, \tag{6.4}$$

unde V este volumul lui S şi $\mathbf{y}$ este un punct din S.

Pe de altă parte, considerăm că avem N eşantioane extrase independent din mulţimea caracterizată prin $p(\mathbf{y})$ şi atunci probabilitatea ca un număr K dintre ele să se găsească în subspaţiul S va fi dată de legea binomială:

$$P(K) = C_N^K P^K (1-P)^{N-K}.$$

Această egalitate se poate justifica astfel: Probabilitatea fiecăruia dintre cele K eşantioane de a se găsi în regiunea S este P, iar probabilitatea fiecăruia dintre celelalte $N-K$ eşantioane de a se găsi în afara acestei regiuni este $1-P$. Deoarece cu cele N eşantioane se pot realiza C_N^K grupuri de câte K, obţinem relaţia de mai sus.

Cunoscând probabilitatea P ca un număr K de eşantioane din totalul celor N să se găsească în domeniul S, în mod evident valoarea medie a lui K este:

$$\epsilon[K] = NP,$$

relaţie ca se poate scrie şi sub forma:

$$\epsilon\left[\frac{K}{N}\right] = P.$$

Dispersia în jurul valorii medii a lui $\frac{K}{N}$ este:

$$\epsilon\left[\left(\frac{K}{N} - P\right)^2\right] = \frac{P(1-P)}{N}$$

şi

$$\epsilon\left[\left(\frac{K}{N} - P\right)^2\right] \to 0 \text{ când } N \to \infty.$$

Aceasta înseamnă că, pentru un număr suficient de mare de eşantioane, valoarea $\frac{K}{N}$ este o estimare corectă a probabilităţii P:

$$P \simeq \frac{K}{N}. \tag{6.5}$$

Din (6.4) şi (6.5) rezultă:

$$p(\mathbf{y})V \simeq \frac{K}{N}$$

sau

$$p(\mathbf{y}) \simeq \frac{K}{NV}. \tag{6.6}$$

Pentru a obţine această estimare a densităţii de probabilitate am făcut două presupuneri referitoare la domeniul S. Am scris relaţia (6.4) presupunând că S este relativ mic astfel încât să putem considera $p(\mathbf{y})$ constantă în interiorul domeniului. Am considerat că S este relativ mare astfel încât să cuprindă un

număr mare de eşantioane pentru a obţine relaţia (6.5). În realitate ne vom găsi între cele două cazuri extreme, de aceea întotdeauna trebuie făcută o alegere care defineşte gradul de precizie al estimării şi, pentru un anumit set de date, există în mod clar o valoare optimă a dimensiunii S astfel încât $p(\mathbf{y})$ să fie estimat cât mai bine posibil.

Atunci când se aplică relaţia (6.6), există două importante opţiuni. Pe de o parte, putem alege o valoare fixă pentru K, determinând volumul V corespunzător pe baza datelor disponibile (*K-nearest neighborhood methods*). Alternativ, putem fixa volumul V, determinând valoarea lui K din setul de date. Această alternativă face parte din clasa tehnicilor de estimare a densităţii de probabilitate folosind nuclee (*kernel-based methods*) şi o vom detalia mai departe.

În condiţiile unui număr infinit de puncte, această procedură de estimare poate deveni exactă pentru că volumul S poate fi restrâns către zero. În felul acesta, ne asigurăm că relaţia (6.4) devine din ce în ce mai precisă în paralel cu creşterea preciziei relaţiei (6.5) prin mărirea numărului de eşantioane din domeniul S. Duda, Hart şi Stork [17] au arătat că ambele abordări descrise mai sus converg către valoarea reală a lui $p(\mathbf{y})$ în condiţiile în care $N \to \infty$, iar V scade odată cu N şi K creşte odată cu N într-un mod convenabil.

Metoda ferestrelor Parzen

Prezentăm metoda de aproximare cu ferestre Parzen a lui $p(\mathbf{y})$ folosind o tehnică bazată pe nuclee [8], [17].

Să presupunem că regiunea S are forma unui hipercub cu muchiile de lungime h, centrat în punctul $\mathbf{y}$. Volumul său va fi:

$$V = h^d. \tag{6.7}$$

Putem găsi o expresie pentru K folosind o funcţie nucleu $W(\mathbf{u})$ numită şi fereastră Parzen care se defineşte astfel:

$$W(\mathbf{u}) = \begin{cases} 1, & \text{când } |u_j| < \frac{1}{2},\ j = 1, \ldots, d \\ 0, & \text{în rest} \end{cases}$$

astfel încât $W(\mathbf{u})$ corespunde hipercubului unitate centrat în origine, iar $\mathbf{u} = [u_1, \ldots, u_d]^t$. În felul acesta, pentru un punct $\mathbf{y}_n$ putem scrie:

$$W\left(\frac{\mathbf{y} - \mathbf{y}_n}{h}\right) = \begin{cases} 1, & \text{dacă } \mathbf{y}_n \text{ se găseşte în interiorul hipercubului} \\ & \text{de muchie } d, \text{ centrat în } \mathbf{y} \\ 0, & \text{în rest.} \end{cases}$$

Folosind această expresie, numărul total de puncte care se găsesc în interiorul hipercubului va fi:

$$K = \sum_{n=1}^{N} W\left(\frac{\mathbf{y} - \mathbf{y}_n}{h}\right). \tag{6.8}$$

Înlocuind (6.7) şi (6.8) în (6.6) obţinem:

$$\widetilde{p(\mathbf{y})} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{h^d} W\left(\frac{\mathbf{y} - \mathbf{y}_n}{h}\right),$$

unde $\widetilde{p(\mathbf{y})}$ este estimatul densităţii de probabilitate.

Deoarece $W(\mathbf{u})$ este o funcţie discontinuă, $\widetilde{p(\mathbf{y})}$ este şi ea o funcţie discontinuă. Se poate evita acest lucru alegând diverse alte forme pentru nucleu.

O alegere foarte utilizată este nucleul normal pentru care:

$$\widetilde{p(\mathbf{y})} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \cdot e^{-\frac{\|\mathbf{y}-\mathbf{y}_n\|^2}{2\sigma^2}}. \tag{6.9}$$

Se foloseşte în mod frecvent notaţia:

$$G(\mathbf{y}, \sigma^2\mathbf{I}) = \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \cdot e^{-\frac{\mathbf{y}^T\mathbf{y}}{2\sigma^2}}, \tag{6.10}$$

unde $G(\mathbf{y}, \sigma^2\mathbf{I})$ este nucleul gaussian (normal) d-dimensional. Mai general, nucleul gaussian într-un spaţiu d-dimensional este definit astfel:

$$G(\mathbf{y}, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{\frac{d}{2}} \|\mathbf{\Sigma}\|^{\frac{1}{2}}} e^{\left(-\frac{1}{2}\mathbf{y}^T\mathbf{\Sigma}^{-1}\mathbf{y}\right)}.$$

Forma (6.10) a nucleului gaussian este una simplificată [8], considerându-se că vectorii de intrare au componente independente şi că pentru fiecare componentă au aceeaşi dispersie. Gaussienele vor fi simetrice, centrate în câte un eşantion $\mathbf{y}$. Astfel, matricea de covarianţă este $\sigma^2\mathbf{I}$, unde $\mathbf{I}$ este matricea unitate. Alegerea nucleului gaussian este motivată de faptul că în dezvoltările care urmează acesta trebuie să fie derivabil pe întregul domeniu.

Estimatorul Parzen al densităţilor de probabilitate înseamnă plasarea câte unui nucleu în locul fiecărui eşantion, iar densitatea este estimată ca medie a valorilor calculate cu aceste nuclee:

$$\widetilde{p(\mathbf{y})} = \frac{1}{N} \sum_{n=1}^{N} G(\mathbf{y} - \mathbf{y}_n, \sigma^2\mathbf{I}). \tag{6.11}$$

Se demonstrează că nucleul gaussian are următoarea proprietate:

$$\int_{-\infty}^{+\infty} G(\mathbf{y} - \mathbf{y}_i, \sigma^2\mathbf{I}) G(\mathbf{y} - \mathbf{y}_j, \sigma^2\mathbf{I}) d\mathbf{y} = G(\mathbf{y}_i - \mathbf{y}_j, 2\sigma^2\mathbf{I}). \tag{6.12}$$

Această relaţie reprezintă convoluţia a două gaussiene care generează tot o gaussiană. Remarcabil este faptul că rezultatul se exprimă ca o interacţiune dintre două eşantioane, iar covarianţa este egală cu suma covarianţelor individuale ale celor două gaussiene. Demonstraţia acestei egalităţi este dată în anexa B.

În general, pentru ca $\widetilde{p(\mathbf{y})} \geqslant 0$ şi $\int_S \widetilde{p(\mathbf{y})} dy = 1$, nucleul trebuie să îndeplinească următoarele două condiţii:

$$W(\mathbf{u}) \geqslant 0$$

$$\int_S W(\mathbf{u}) d\mathbf{u} = 1.$$

În relaţia (6.9), σ joacă rolul unui parametru care determină calitatea estimării. Atunci când σ este mare, netezirea lui $\widetilde{p(\mathbf{y})}$ este foarte accentuată, pierzându-se eventualele maxime locale. La extrema cealaltă, când $\sigma \to 0$ nucleul se apropie de funcţia delta, graficul lui $\widetilde{p(\mathbf{y})}$ conţine variaţii foarte dese. Din acest motiv, trebuie găsit un compromis între cele două variante, iar valoarea lui σ poate fi stabilită experimental.

Entropia Renyi

Estimarea densităţii de probabilitate cu ferestre Parzen oferă o metodă de estimare a entropiei cu avantaje certe din punct de vedere computaţional atunci când se foloseşte împreună cu entropia Renyi [84]. Această măsură alternativă a entropiei este, de fapt, o familie de măsuri care include entropia Shannon ca un caz particular [26], [79]. Pentru o variabilă aleatoare discretă C, entropia Renyi se scrie:

$$H_{R_\alpha}(C) = \frac{1}{1-\alpha} \log \sum_c p^\alpha(c), \tag{6.13}$$

iar pentru o variabilă aleatoare continuă Y entropia Renyi se defineşte prin:

$$H_{R_\alpha}(Y) = \frac{1}{1-\alpha} \log \int_{\mathbf{y}} p^\alpha(\mathbf{y}) d\mathbf{y}, \tag{6.14}$$

unde $\alpha > 0$ şi $\alpha \neq 1$.

Este cunoscută relaţia care există între entropia Shannon şi entropia Renyi:

$$\lim_{\alpha \to 1} H_{R_\alpha} = H.$$

Această egalitate se poate verifica imediat după cum urmează. Pentru cazul discret avem:

$$\sum_c p(c) = 1$$

şi atunci putem scrie:

$$\lim_{\alpha \to 1} H_{R_\alpha}(C) = \lim_{\alpha \to 1} \frac{\log \sum_c p^\alpha(c)}{1-\alpha}.$$

Deoarece avem cazul de nedeterminare $\frac{0}{0}$, aplicăm regula lui l'Hospital şi obţinem:

$$\begin{aligned} \lim_{\alpha \to 1} H_{R_\alpha}(C) &= \lim_{\alpha \to 1} \left(-\frac{1}{\sum_c p^\alpha(c)} \sum_c p^\alpha(c) \log p(c) \right) = \\ &= -\sum_c p(c) \log p(c), \end{aligned} \tag{6.15}$$

adică relaţia dintre entropia Renyi şi entropia Shannon pentru cazul discret. Atunci când entropia Renyi se defineşte pentru o variabilă aleatoare continuă, demonstraţia este similară. Avem:

$$\int_{\mathbf{y}} p(\mathbf{y}) d\mathbf{y} = 1$$

şi atunci obţinem:

$$\lim_{\alpha \to 1} H_{R_\alpha}(Y) = \lim_{\alpha \to 1} \frac{\log \int_{\mathbf{y}} p^\alpha(\mathbf{y}) d\mathbf{y}}{1-\alpha}.$$

Aplicăm regula lui l'Hospital pentru cazul de nedeterminare $\frac{0}{0}$ şi obţinem relaţia dintre entropia Renyi şi entropia Shannon în cazul continuu:

$$\lim_{\alpha \to 1} H_{R_\alpha}(Y) = -\int_{\mathbf{y}} p(\mathbf{y}) \log p(\mathbf{y}).$$

Pentru $\alpha = 2$ avem entropia Renyi pătratică. În cazul discret, relaţia (6.13) devine:

$$H_{R_2}(C) = -\log \sum_c p^2(c),$$

iar pentru cazul continuu relaţia (6.14) se scrie:

$$H_{R_2}(Y) = -\log \int_{\mathbf{y}} p^2(\mathbf{y}) d\mathbf{y}. \tag{6.16}$$

Pentru dezvoltarea algoritmilor MIRLVQ şi ERLVQ vom fi interesaţi doar de cazul continuu. Din relaţia (6.11), proprietatea (6.12) şi expresia (6.16) obţinem:

$$\begin{aligned} H_{R_2}(Y) &= -\log \frac{1}{N^2} \int_{\mathbf{y}} \left(\sum_{k=1}^{N} \sum_{j=1}^{N} G(\mathbf{y} - \mathbf{y}_k, \sigma^2 \mathbf{I}) G(\mathbf{y} - \mathbf{y}_j, \sigma^2 \mathbf{I}) \right) d\mathbf{y} = \\ &= -\log \frac{1}{N^2} \sum_{k=1}^{N} \sum_{j=1}^{N} G(\mathbf{y}_k - \mathbf{y}_j, 2\sigma^2 \mathbf{I}) \end{aligned} \tag{6.17}$$

după ce integrala a comutat cu cele două sume.

Am arătat astfel că estimarea densităţii de probabilitate cu ferestre Parzen este utilă pentru estimarea entropiei Renyi pătratice ca sumă de interacţiuni locale între toate perechile de eşantioane. În secţiunea următoare vom folosi aceste rezultate pentru estimarea informaţiei mutuale dintre două variabile aleatoare.

6.4 Informaţia mutuală pătratică

Informaţia mutuală este o măsură a dependenţei dintre două variabile aleatoare şi se defineşte prin următoarea diferenţă de entropii Shannon:

$$I(X,Y) = H(Y) - H(Y|X),$$

unde X şi Y sunt două variabile aleatoare.

În [53] se fac o serie de consideraţii referitoare la informaţia mutuală si la funcţia de corelaţie care măsoară, de asemenea, dependenţa dintre două variabile aleatoare. Reluăm în paragraful următor, pe scurt, aceste observaţii.

Dacă cele două variabile aleatoare sunt independente, atunci informaţia mutuală dintre ele este nulă. Când cele două variabile aleatoare sunt dependente, de exemplu una se poate calcula în funcţie de cealaltă, informaţia mutuală dintre ele este mare. Informaţia mutuală arată care este gradul de predictibilitate a unei variabile aleatoare atunci când cealaltă este cunoscută. Interpretările de mai sus ale informaţiei mutuale sunt proprii şi corelaţiei. Cu toate acestea, informaţia mutuală măsoară dependenţa generală dintre două variabile aleatoare, în timp ce corelaţia măsoară dependenţa liniară. O altă diferenţă majoră dintre cele două măsuri este aceea că informaţia mutuală poate fi aplicată atât secvenţelor simbolice cât şi secvenţelor numerice, iar corelaţia poate fi aplicată doar secvenţelor numerice. Problema pe care o discutăm în acest capitol asimilează colecţia de clase de forme unei secvenţe simbolice, de aceea informaţia mutuală este o alegere naturală.

Deoarece entropia Shannon nu poate fi estimată din datele de instruire cu un efort computaţional rezonabil, folosim o soluţie alternativă de estimare a informaţiei mutuale cu ajutorul divergenţei Kulback-Leibler (KL) care se mai numeşte şi entropie relativă. Divergenţa KL dintre două funcţii de densitate de probabilitate $f(\mathbf{x})$ şi $g(\mathbf{x})$ se defineşte prin [17]:

$$K(f,g) = \int f(\mathbf{x}) \log \frac{f(\mathbf{x})}{g(\mathbf{x})} d\mathbf{x} \tag{6.18}$$

care foloseşte implicit entropia Shannon şi satisface inegalitatea lui Gibbs [54]:

$$K(f,g) \geqslant 0,$$

egalitatea având loc pentru:

$$f(\mathbf{x}) = g(\mathbf{x}), \forall \mathbf{x} \in X.$$

Informaţia mutuală dintre două variabile aleatoare X şi Y se estimează prin divergenţa KL astfel:

$$\begin{aligned} I_S(X,Y) &= K(f_{XY}(\mathbf{x},\mathbf{y}), f_X(\mathbf{x})f_Y(\mathbf{y})) = \\ &= \int\int f_{XY}(\mathbf{x},\mathbf{y}) \log \frac{f_{XY}(\mathbf{x},\mathbf{y})}{f_X(\mathbf{x})f_Y(\mathbf{y})} d\mathbf{x}d\mathbf{y}. \end{aligned}$$

Dorim să estimăm informaţia mutuală dintre două variabile aleatoare într-o manieră similară celei în care am estimat entropia pătratică Renyi. Relaţia de mai sus nu poate fi, însă, integrată uşor cu metoda de estimare a densităţilor de probabilitate cu ferestre Parzen deoarece nu putem aplica proprietatea (6.12). Pentru aceasta ar trebui ca toţi termenii să aibă gradul doi.

Din acest motiv, în [70] se propune o nouă măsură a distanţei dintre două funcţii de densitate de probabilitate care conţine termeni de gradul doi şi care păstrează proprietăţi ale divergenţei KL. Aceată măsură porneşte de la formula estimării entropiei Renyi şi este derivată euristic menţinând termenii de gradul doi, similar entropiei Renyi pătratice.

Pe baza distanţei euclidiene se poate scrie următoarea inegalitate:

$$(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y}) \geqslant 0 \Leftrightarrow \|\mathbf{x}\|^2 + \|\mathbf{x}\|^2 - 2\mathbf{x}^T\mathbf{y} \geqslant 0$$

şi atunci rezultă următoarea estimare a divergenţei dintre $f(\mathbf{x})$ şi $g(\mathbf{x})$:

$$I_{ED}(f,g) = \int f^2(\mathbf{x})d\mathbf{x} + \int g^2(\mathbf{x})d\mathbf{x} - 2\int f(\mathbf{x})g(\mathbf{x})d\mathbf{x}.$$

Principe, Xu şi Fisher [70] au arătat că această nouă măsură are aceleaşi minime şi maxime ca şi divergenţa KL, de aceea o poate înlocui pe cea din (6.18) în probleme care folosesc optimizarea informaţiei mutuale.

Atunci informaţia mutuală exprimată pe baza acestei noi măsuri este divergenţa dintre $p(\mathbf{x},\mathbf{y})$ şi $p(\mathbf{x})p(\mathbf{y})$:

$$\begin{aligned} I_{ED}(X,Y) &= \int_{\mathbf{x}}\int_{\mathbf{y}} p^2(\mathbf{x},\mathbf{y})d\mathbf{x}d\mathbf{y} + \int_{\mathbf{x}}\int_{\mathbf{y}} p^2(\mathbf{x})p^2(\mathbf{y})d\mathbf{x}d\mathbf{y} - \\ &\quad -2\int_{\mathbf{x}}\int_{\mathbf{y}} p(\mathbf{x},\mathbf{y})p(\mathbf{x})p(\mathbf{y})d\mathbf{x}d\mathbf{y}. \end{aligned} \tag{6.19}$$

În mod evident, $I_{ED}(X,Y) \geqslant 0$ şi egalitatea are loc dacă şi numai dacă X şi Y sunt statistic independente, cu alte cuvinte aceste două proprietăţi ale divergenţei KL se păstrează şi pentru $I_{ED}(X,Y)$.

Torkkola introduce în [84] aceeaşi formulă de estimare a informaţiei mutuale folosind, însă, o argumentaţie diferită pe care o prezentăm în continuare.

Aşa cum am văzut, informaţia mutuală poate fi privită ca o divergenţă KL dintre $p(c, \mathbf{y})$ şi produsul $p(c)p(\mathbf{y})$. Există o serie de măsuri alternative ale divergenţei care pot fi utilizate pentru scopul propus în această secţiune. Una dintre acestea este divergenţa propusă de Kapur [84] pentru două distribuţii discrete P şi Q:

$$D(P,Q) = \frac{1}{\alpha(\alpha-1)} \sum_{i=1}^{n} \left[p_i^{\alpha} - \alpha p_i q_i^{\alpha-1} + (\alpha-1) q_i^{\alpha} \right], \ \alpha \neq 0, \alpha \neq 1.$$

Pentru $\alpha = 2$, ignorând factorul $\frac{1}{2}$ şi extinzând măsura pentru densităţi continue, se obţine:

$$D(f,g) = \int_{\mathbf{x}} (f(\mathbf{x}) - g(\mathbf{x}))^2 d\mathbf{x}. \tag{6.20}$$

Această măsură este întotdeauna pozitivă şi ia valoarea 0 atunci când $f(\mathbf{x}) = g(\mathbf{x})$, acestea fiind şi proprietăţile divergenţei KL. Kapur nu dă, însă, o justificare formală a condiţiilor în care relaţia de mai sus este adevărată [84].

O altă justificare a folosirii măsurii pătratice este dată tot de Torkkola în [84], bazându-se următoarea divergenţă:

$$V(f,g) = \int_{\mathbf{x}} |f(\mathbf{x}) - g(\mathbf{x})| d\mathbf{x}.$$

Acelaşi autor arată că inegalitatea lui Pinsker dă o limită inferioară pentru $K(f,g)$:

$$K(f,g) \geqslant \frac{1}{2} V(f,g)^2. \tag{6.21}$$

Pe de altă parte, din cauză că $f(\mathbf{x})$ şi $g(\mathbf{x})$ sunt densităţi de probabilitate, iau valori în intervalul $[0,1]$ şi:

$$|f(\mathbf{x}) - g(\mathbf{x})| \geqslant (f(\mathbf{x}) - g(\mathbf{x}))^2,$$

de unde rezultă că:

$$V(f,g) \geqslant D(f,g). \tag{6.22}$$

Din inegalităţile (6.21) şi (6.22) rezultă că a îl maximiza pe $D(f,g)$ este echivalent cu a maximiza limita inferioară a lui $K(f,g)$.

Am văzut că pentru două variabile aleatoare continue Y_1 şi Y_2 informaţia mutuală se exprimă ca divergenţă între $p(\mathbf{y}_1, \mathbf{y}_2)$ şi produsul $p(\mathbf{y}_1)p(\mathbf{y}_2)$. Introducând aceste valori în (6.20) rezultă:

$$I_T(Y_1, Y_2) = \int \int (p(\mathbf{y}_1, \mathbf{y}_2) - p(\mathbf{y}_1)p(\mathbf{y}_2))^2 d\mathbf{y}_1 d\mathbf{y}_2$$

care este echivalentă cu I_{ED}.

Torkkola [83] a arătat că această relaţie dezvoltată pentru două variabile aleatoare continue poate fi adaptată pentru a obţine o estimare a informaţiei mutuale dintre o variabilă aleatoare discretă C şi o variabilă aleatoare continuă Y, caz în care (6.19) devine:

$$\begin{aligned} I_{ED}(C,Y) &= \sum_c \int_{\mathbf{y}} p^2(c,\mathbf{y})d\mathbf{y} + \sum_c \int_{\mathbf{y}} p^2(c)p^2(\mathbf{y})d\mathbf{y} - \\ &\quad -2\sum_c \int_{\mathbf{y}} p(c,\mathbf{y})p(c)p(\mathbf{y})d\mathbf{y}. \end{aligned} \tag{6.23}$$

Notăm cu M_p numărul de eşantioane din clasa p, cu $\mathbf{y}_k$ eşantionul numărul k din întreg setul de eşantioane disponibile fără a ne interesa clasa căreia îi aparţine şi cu $\mathbf{y}_{pj}$ acelaşi eşantion arătând că el aparţine clasei p şi că are indicele j în clasă. Folosim următoarele egalităţi cunoscute:

$$p(c_p,\mathbf{y}) = P(c_p)p(\mathbf{y}|c_p)$$

şi

$$p(\mathbf{y}) = \sum_{p=1}^{M} p(c_p,\mathbf{y}),$$

unde M este numărul de clase.

Rescriem (6.23) folosind aproximările densităţilor de probabilitate cu ferestre Parzen şi rezultă:

$$\begin{aligned} I_{ED}(C,Y) &= \frac{1}{N^2}\sum_{p=1}^{M}\sum_{k=1}^{M_p}\sum_{l=1}^{M_p} G(\mathbf{y}_{pk}-\mathbf{y}_{pl},2\sigma^2\mathbf{I}) + \\ &+ \frac{1}{N^2}\left(\sum_{p=1}^{M}\left(\frac{M_p}{N}\right)^2\right)\sum_{k=1}^{N}\sum_{l=1}^{N} G(\mathbf{y}_k-\mathbf{y}_l,2\sigma^2\mathbf{I}) - \\ &- 2\frac{1}{N^2}\sum_{p=1}^{M}\frac{M_p}{N}\sum_{j=1}^{M_p}\sum_{k=1}^{N} G(\mathbf{y}_{pj}-\mathbf{y}_k,2\sigma^2\mathbf{I}), \end{aligned} \tag{6.24}$$

unde N reprezintă numărul total de eşantioane.

6.5 Algoritmul Mutual Information Relevance LVQ (MIRLVQ)

În această secţiune introducem o nouă metodă de instruire a importanţei componentelor vectorilor de intrare, obţinută prin maximizarea informaţiei mutuale dintre etichetele claselor şi o transformare a vectorilor de intrare. Aşa

cum am arătat în secţiunea precedentă, informaţia mutuală pătratică este o sumă de termeni de gradul doi, fiind dezvoltată pornind de la similaritatea cu entropia pătratică Renyi. Informaţia mutuală pătratică estimată cu ferestre Parzen oferă avantajul că se calculează ca sumă de gaussiene, fiecare dintre ele putând fi evaluată folosind perechi de eşantioane din setul de instruire.

Considerăm că avem o variabilă aleatoare X, iar $\mathbf{x}_i$, $i = 1, \ldots, N$ este un set de realizări ale acestei variabile aleatoare. Realizările vor fi chiar vectorii de intrare. Considerăm o variabilă aleatoare discretă C şi c_j, $j = 1, \ldots, M$ sunt realizările sale care reprezintă etichetele claselor din care fac parte vectorii $\mathbf{x}_i$. Pentru ca instruirea să fie eficientă, trebuie ca fiecărei clase să i se asocieze mai mulţi vectori de intrare.

Algoritmul Weighted LVQ

Am arătat în capitolul 5 că actualizarea factorilor de relevanţă prin algoritmii DSLVQ şi RLVQ este motivată euristic. Din acest motiv, am introdus algoritmul OWA-RLVQ care calculează relevanţele ca ponderi ale unui operator OWA prin minimizarea distanţei modificate

$$D_{ij}^{*} = \sqrt{\sum_{k=1}^{n} \lambda_k (|x_{ik} - w_{jk}|^{*})^2},$$

unde notaţiile sunt cele din secţiunea 5.4. Algoritmul MIRLVQ va folosi metrica ponderată

$$D_{ij} = \|\mathbf{x}_i - \mathbf{w}_j\|_{\lambda} = \sqrt{\sum_{k=1}^{n} \lambda_k (x_{ij} - w_{jk})^2}$$

care a fost introdusă în algoritmul RLVQ.

Includem aceste variaţii ale lui LVQ într-o clasă generală de algoritmi pe care o numim *Weighted LVQ*. Un algoritm *weighted LVQ* are următoarele caracteristici:

a) Foloseşte o distanţă ponderată;

b) Relevanţele şi prototipurile sunt actualizate simultan în timpul fazei de instruire. Actualizarea poate fi realizată incremental după prelucrarea fiecărui vector de instruire;

c) Relevanţele care se obţin după încheierea algoritmului de instruire pot fi privite ca ranguri ale componentelor vectorilor de intrare şi se pot utiliza pentru reducerea spaţiului intrărilor (*feature selection*).

Un algoritm *weighted LVQ* constă din următorii paşi:

Pasul 1. *Găseşte cel mai apropiat prototip* $\mathbf{w}_j$*, câştigătorul, pentru care valoarea lui* $\|\mathbf{x}_i - \mathbf{w}_j\|_\lambda$ *este minimă pentru întreagă colecţia de vectori prototip.*

Pasul 2. *Actualizează prototipul* $\mathbf{w}_j$ *câştigător după un algoritm LVQ.*

Pasul 3. *Actualizează vectorul relevanţelor* λ*.*

Pasul 4. *Actualizează rangul fiecărei componente a vectorilor de intrare ca o medie a paşilor anteriori.*

Ne propunem să transpunem algoritmii de tip *weighted LVQ* într-un cadru teoretic diferit. Introducem algoritmul MIRLVQ ca un algoritm din această clasă bazat pe maximizarea informaţiei mutuale. O problemă majoră pe care o ridică acestă abordare este marea complexitate computaţională pe care o implică estimarea informaţiei mutuale. Pentru a evita acest dezavantaj, vom folosi metoda de estimare a informaţiei mutuale cu ferestre Parzen.

Adaptarea factorilor de relevanţă

Notăm

$$\mathbf{y}_i = \lambda \mathbf{I}(\mathbf{x}_i - \mathbf{w}_j), \tag{6.25}$$

unde $\mathbf{x}_i$ este vectorul de intrare i, $\mathbf{w}_j$ este prototipul j obţinut cu un algoritm din categoria weighted LVQ, λ este vectorul relevanţelor şi $\mathbf{y}_i$ este vectorul de ieşire. Aceasta este transformarea pe care dorim să o implementăm, λ fiind un vector global care caracterizează toate perechile intrare-prototip.

Considerăm informaţia mutuală

$$I(c_j; \mathbf{y}_i) = I(c_j; \mathbf{x}_i, \mathbf{w}_j, \lambda),$$

unde $\mathbf{y}_i$ sunt realizări ale variabilei aleatoare continue Y. În relaţia anterioară, am folosit două notaţii care au aceeaşi semnificaţie cu $I(C, Y)$. Această egalitate înglobează ipoteza formulată în relaţia (6.25), conform căreia ieşirea sistemului nostru se calculează folosind un vector de intrare $\mathbf{x}_i$, un prototip $\mathbf{w}_j$ asociat lui calculat printr-un algoritm tip weighted LVQ şi vectorul relevanţelor λ care este comun tuturor vectorilor din setul de instruire. Atunci când vectorul de intrare $\mathbf{x}_i$ este asemănător prototipului $\mathbf{w}_j$, componentele vectorului λ trebuie întărite. Putem interpreta această similaritate prin faptul că informaţia mutuală dintre clasa c_j şi valoarea agregată $\mathbf{y}_i$ este mare. Pornind de la această observaţie, putem calcula relevanţele prin maximizarea informaţiei mutuale şi

folosim tehnica gradientului crescător. La fiecare pas de instruire $t+1$ vom avea:

$$\lambda^{(t+1)} = \lambda^{(t)} + \alpha \frac{\partial I}{\partial \lambda} \Leftrightarrow$$

$$\lambda^{(t+1)} = \lambda^{(t)} + \alpha \sum_{i=1}^{n} \frac{\partial I}{\partial \mathbf{y}_{ij}} \mathbf{I} \frac{\partial \mathbf{y}_{ij}}{\partial \lambda}. \tag{6.26}$$

Vectorii $\mathbf{x}_i$ şi $\mathbf{w}_j$ nu sunt dependenţi de λ, de aceea:

$$\frac{\partial \mathbf{y}_i}{\partial \lambda} = \mathbf{x}_i - \mathbf{w}_j$$

şi (6.26) devine:

$$\lambda^{(t+1)} = \lambda^{(t)} + \alpha \sum_{i=1}^{n} \frac{\partial I}{\partial \mathbf{y}_i} \mathbf{I} \left(\mathbf{x}_i - \mathbf{w}_j \right). \tag{6.27}$$

Trebuie să estimăm acum

$$\frac{\partial I}{\partial \mathbf{y}_i}$$

şi pentru aceasta folosim măsura informaţiei mutuale dintre o variabilă aleatoare continuă şi una discretă dată de relaţia (6.23):

$$I(C,Y) = \sum_{j=1}^{M} \int_{\mathbf{y}} p^2(c_j, \mathbf{y}) d\mathbf{y} + \sum_{j=1}^{M} \int_{\mathbf{y}} p^2(c_j) p^2(\mathbf{y}) d\mathbf{y} -$$

$$-2 \sum_{j=1}^{M} \int_{\mathbf{y}} p(c_j, \mathbf{y}) p(c_j) p(\mathbf{y}) d\mathbf{y}.$$

Utilizăm relaţia probabilităţilor condiţionate:

$$p(\mathbf{y}, c_j) = P(c_j) p(\mathbf{y}|c_j),$$

unde $P(c_j)$ este probabilitatea *a priori* a clasei c_j, $p(\mathbf{y}|c_j)$ este densitatea de probabilitate condiţionată a vectorului $\mathbf{y}$ care provine dintr-un vector de intrare ce aparţine clasei c_j, iar

$$p(\mathbf{y}) = \sum_{j=1}^{M} p(c_j, \mathbf{y})$$

este densitatea de probabilitate a eşantionului $\mathbf{y}$. Analizând formula anterioară, trebuie să facem observaţia că un $\mathbf{y}$ particular poate fi generat de diverşi vectori de intrare $\mathbf{x}_i$ care pot aparţine oricărei clase. Probabilitatea condiţionată arată clasa din care a provenit vectorul de intrare care a generat respectiva ieşire, fiind elementul care stabileşte legătura dintre clasă şi vectorul de ieşire.

Calculul gradientului aşa cum este descris în formulele anterioare poate fi înlocuit cu un algoritm mult mai simplu, de gradient stohastic. Se folosesc două eşantioane aleatoare $\mathbf{x}_1$ şi $\mathbf{x}_2$ reprezentative pentru întreaga bază de date [83].

Prin estimarea informaţiei mutuale cu ferestre Parzen, din (6.24) se pot face ajustări ale lui λ în direcţia gradientului pozitiv.

Vom trata separat următoarele două cazuri: cel în care eşantioanele sunt din aceeaşi clasă şi cel în care eşantioanele sunt din clase diferite.

a) În prima situaţie, putem considera că pentru clasa reprezentată de cele două eşantioane avem probabilitatea *a priori* $P(c_1) = 1$, iar pentru celelalte avem $P(c_k) = 0$, $k \neq 1$. În relaţia (6.24) vom avea $N = 2$, $M = 2$, $M_1 = 2$, $M_2 = 0$ şi vom scrie:

$$\begin{aligned}
I(C,Y) &= \frac{1}{2^2}\sum_{p=1}^{2}\sum_{k=1}^{M_p}\sum_{l=1}^{M_p} G(\mathbf{y}_{pk}-\mathbf{y}_{pl}, 2\sigma^2\mathbf{I}) + \\
&+ \frac{1}{2^2}\left(\sum_{p=1}^{2}\left(\frac{M_p^2}{2}\right)\right)\sum_{k=1}^{2}\sum_{l=1}^{2} G(\mathbf{y}_k-\mathbf{y}_l, 2\sigma^2\mathbf{I}) - \\
&- 2\cdot\frac{1}{2^2}\cdot\sum_{p=1}^{2}\frac{M_p}{2}\sum_{j=1}^{M_p}\sum_{k=1}^{2} G(\mathbf{y}_{pj}-\mathbf{y}_k, 2\sigma^2\mathbf{I}) = \\
&= \frac{1}{4}\left[G(\mathbf{y}_{11}-\mathbf{y}_{11}, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_{11}-\mathbf{y}_{12}, 2\sigma^2\mathbf{I}) + \right. \\
&\quad \left. + G(\mathbf{y}_{12}-\mathbf{y}_{11}, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_{12}-\mathbf{y}_{12}, 2\sigma^2\mathbf{I})\right] + \\
&+ \frac{1}{4}\cdot 1\cdot\left[G(\mathbf{y}_1-\mathbf{y}_1, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_1-\mathbf{y}_2, 2\sigma^2\mathbf{I}) + \right. \\
&\quad \left. + G(\mathbf{y}_2-\mathbf{y}_1, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_2-\mathbf{y}_2, 2\sigma^2\mathbf{I})\right] - \\
&- 2\cdot\frac{1}{4}\cdot\left[G(\mathbf{y}_{11}-\mathbf{y}_1, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_{11}-\mathbf{y}_2, 2\sigma^2\mathbf{I}) + \right. \\
&\quad \left. + G(\mathbf{y}_{12}-\mathbf{y}_1, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_{12}-\mathbf{y}_2, 2\sigma^2\mathbf{I})\right].
\end{aligned}$$

Această relaţie se poate simplifica ţinând cont de observaţia că

$$\mathbf{y}_{11} = \mathbf{y}_1$$

şi

$$\mathbf{y}_{12} = \mathbf{y}_2,$$

justificate prin faptul că toate eşantioanele fac parte din aceeaşi clasă, cea cu indicele 1. În aceste condiţii, informaţia mutuală aproximată cu ferestre Parzen conform relaţiei (6.24) devine:

$$\begin{aligned}
I(C,Y) &= \frac{1}{2}\left[G(\mathbf{y}_1-\mathbf{y}_2, 2\sigma^2\mathbf{I}) + G(0, 2\sigma^2\mathbf{I})\right] + \\
&+\frac{1}{2}\left[G(\mathbf{y}_1-\mathbf{y}_2, 2\sigma^2\mathbf{I}) + G(0, 2\sigma^2\mathbf{I})\right] - \\
&-2\cdot\frac{1}{2}\cdot\left[G(\mathbf{y}_1-\mathbf{y}_2, 2\sigma^2\mathbf{I}) + G(0, 2\sigma^2\mathbf{I})\right],
\end{aligned}$$

adică

$$I(C,Y) = 0.$$

Aşadar, atunci când cele două eşantioane aparţin aceleiaşi clase, informaţia mutuală calculată mai sus este nulă. Introducând acest rezultat în (6.27) rezultă că:

$$\lambda^{(t+1)} = \lambda^{(t)},$$

însemnând că asupra relevanţelor nu apare nici o modificare.

b) A doua situaţie este cea în care cele două eşantioane sunt din clase diferite şi atunci probabilităţile *a priori* ale claselor din care acestea fac parte sunt $P(c_1) = \frac{1}{2}$ şi $P(c_2) = \frac{1}{2}$, iar $P(c_k) = 0$ pentru $k \notin \{1,2\}$. În relaţia (6.24) vom avea $N = 2$, $M = 2$, $M_1 = 1$, $M_2 = 1$ şi vom scrie:

$$\begin{aligned}
I(C,Y) &= \frac{1}{2^2}\sum_{p=1}^{2}\sum_{k=1}^{M_p}\sum_{l=1}^{M_p} G(\mathbf{y}_{pk} - \mathbf{y}_{pl}, 2\sigma^2\mathbf{I}) + \\
&+ \frac{1}{2^2}\left(\sum_{p=1}^{2}\left(\frac{M_p^2}{2}\right)\right)\sum_{k=1}^{2}\sum_{l=1}^{2} G(\mathbf{y}_k - \mathbf{y}_l, 2\sigma^2\mathbf{I}) - \\
&- 2\cdot\frac{1}{2^2}\cdot\sum_{p=1}^{2}\frac{M_p}{2}\sum_{j=1}^{M_p}\sum_{k=1}^{2} G(\mathbf{y}_{pj} - \mathbf{y}_k, 2\sigma^2\mathbf{I}) = \\
&= \frac{1}{4}\left[G(\mathbf{y}_{11} - \mathbf{y}_{11}, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_{21} - \mathbf{y}_{21}, 2\sigma^2\mathbf{I})\right] + \\
&+ \frac{1}{4}\cdot\frac{1}{4}\cdot 2\cdot\left[G(\mathbf{y}_1 - \mathbf{y}_1, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I}) + \right. \\
&\quad \left. + G(\mathbf{y}_2 - \mathbf{y}_1, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_2 - \mathbf{y}_2, 2\sigma^2\mathbf{I})\right] - \\
&- 2\cdot\frac{1}{4}\cdot\frac{1}{2}\left[G(\mathbf{y}_{11} - \mathbf{y}_1, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_{11} - \mathbf{y}_2, 2\sigma^2\mathbf{I}) + \right. \\
&\quad \left. + G(\mathbf{y}_{21} - \mathbf{y}_1, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_{21} - \mathbf{y}_2, 2\sigma^2\mathbf{I})\right]. \qquad (6.28)
\end{aligned}$$

Remarcăm că $\mathbf{y}_{11} = \mathbf{y}_{21} = \mathbf{y}_1$, şi de această dată relaţia (6.28) se va scrie:

$$\begin{aligned}
I(C,Y) &= \frac{1}{2}G(0, 2\sigma^2\mathbf{I}) + \frac{1}{4}\left[G(0, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})\right] \\
&\quad -2\frac{1}{4}\left[G(0, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})\right] = \\
&= \frac{1}{4}\left[G(0, 2\sigma^2\mathbf{I}) - G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})\right]. \qquad (6.29)
\end{aligned}$$

Având acest rezultat, putem stabili acum care este relaţia de actualizare a factorilor de relevanţă prin introducerea lui (6.29) în (6.27). Trebuie să calculăm mai întâi:

$$\frac{\partial I(C,Y)}{\partial \mathbf{y}_1} = \frac{\partial}{\partial \mathbf{y}_1}\left(\frac{1}{4}\left(G(0, 2\sigma^2\mathbf{I}) - G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})\right)\right) =$$

$$
\begin{aligned}
&= -\frac{1}{4}\frac{\partial G(0, 2\sigma^2\mathbf{I})}{\partial \mathbf{y}_1} - \frac{1}{4}\frac{\partial G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})}{\partial \mathbf{y}_1} = \\
&= -\frac{1}{8\sigma^2}\frac{\partial G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})}{\partial \mathbf{y}_1}
\end{aligned} \tag{6.30}
$$

şi

$$
\begin{aligned}
\frac{\partial I(C,Y)}{\partial \mathbf{y}_2} &= \frac{\partial}{\partial \mathbf{y}_2}\left(\frac{1}{4}\left(G(0, 2\sigma^2\mathbf{I}) - G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})\right)\right) = \\
&= -\frac{1}{4}\frac{\partial G(0, 2\sigma^2\mathbf{I})}{\partial \mathbf{y}_2} - \frac{1}{4}\frac{\partial G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})}{\partial \mathbf{y}_2} = \\
&= \frac{1}{8\sigma^2}\frac{\partial G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})}{\partial \mathbf{y}_1},
\end{aligned} \tag{6.31}
$$

unde am folosit faptul că

$$
G(0, 2\sigma^2\mathbf{I}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\sigma|^{\frac{1}{2}}}
$$

este o constantă şi derivata ei este nulă. Din (6.27), (6.30) şi (6.31) rezultă că actualizarea factorilor de relevanţă se face astfel:

$$
\begin{aligned}
\lambda^{(t+1)} &= \lambda^{(t)} - \frac{\alpha}{8\sigma^2} G\left(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I}\right)(\mathbf{y}_2 - \mathbf{y}_1)\,\mathbf{I}\left[\left(\mathbf{x}_1 - \mathbf{w}_{j(1)}\right) - \right. \\
&\quad \left. - \left(\mathbf{x}_2 - \mathbf{w}_{j(2)}\right)\right],
\end{aligned} \tag{6.32}
$$

unde prin $\mathbf{w}_{j(1)}$ şi $\mathbf{w}_{j(2)}$ am notat prototipurile câştigătoare pentru vectorii $\mathbf{x}_1$ şi $\mathbf{x}_2$.

Am stabilit această regulă de modificare iterativă a relevanţelor prin maximizarea informaţiei mutuale $I(C, Y)$. Vom dezvolta în secţiunile 6.6 şi 6.7 o metodă similară de calcul al factorilor de relevanţă prin maximizarea unei mărimi bazate pe energia informaţională a variabilelor aleatoare. În final, vom detalia algoritmul care calculează în paralel atât prototipurile $\mathbf{w}_j$ cât şi relevanţele λ.

6.6 Energia informaţională

Prezentăm în această secţiune definiţiile şi cele mai importante proprietăţi ale energiei informaţionale care este o mărime introdusă de O. Onicescu în [65].

Pentru variabila aleatoare continuă X, energia informaţională se defineşte astfel:

$$
E(X) = \int_{-\infty}^{+\infty} p^2(\mathbf{x})d\mathbf{x},
$$

unde $p(\mathbf{x})$ este funcţia de densitate de probabilitate a variabilei aleatoare.

În cazul în care avem o variabilă aleatoare discretă C, energia informaţională se defineşte prin:

$$E(C) = \sum_{i=1}^{n} p^2(c_i),$$

unde distribuţia aleatoare este definită prin

$$p(c_k) \geqslant 0$$

şi

$$\sum_{k=1}^{n} p(c_k) = 1.$$

Dată fiind această definiţie, următorul rezultat este adevărat [71]:

Propoziţie. *Pentru orice repartiţie* $p(c_k) \geqslant 0$, $\sum_{k=1}^{n} p(c_k) = 1$, *între energia informaţională şi entropia Renyi are loc relaţia*

$$H_{R_2}(C) = -\log E(C)$$

sau

$$E(C) = 2^{-H_{R_2}(C)}.$$

În consecinţă, entropia pătratică Renyi şi energia informaţională se determină reciproc şi în mod unic.

Vom discuta proprietăţile energiei informaţionale discrete, iar cazul continuu este similar.

Considerăm o altă variabilă aleatoare discretă T pentru care avem probabilităţile $p(t_k) \geqslant 0$, $k = 1, \ldots, m$ corespunzătoare celor m realizări posibile şi $\sum_{k=1}^{m} p(t_k) = 1$. Atunci:

$$p(c_i, t_k) = p(t_k)p(c_i|t_k) = p(c_i)p(t_k|c_i).$$

Variabilele aleatoare C şi T sunt independente dacă şi numai dacă $p(c_i, t_k) = p(c_i)p(t_k)$.

Energia informaţională medie a variabilei C condiţionată de T se defineşte astfel:

$$E(C|T) = \sum_{k=1}^{m} p(t_k)E(C|t_k),$$

unde

$$E(C|t_k) = \sum_{i=1}^{n} p^2(c_i|t_k).$$

Cu aceste relaţii, energia informaţională condiţionată se mai poate scrie:

$$\begin{aligned} E(C|T) &= \sum_{i=1}^{n}\sum_{k=1}^{m} p(t_k)p^2(c_i|t_k) = \\ &= \sum_{i=1}^{n}\sum_{k=1}^{m} p(c_i,t_k)p(c_i|t_k). \end{aligned}$$

În [2] se arată că există două strategii care se pot adopta atunci când se studiază relaţiile dintre două sisteme care interacţionează: se poate măsura interdependenţa acestora ca atribut mutual sau se poate cuantifica măsura în care un sistem depinde de celălalt.

Prima strategie a fost abordată de S. Watanabe care a folosit entropia Shannon pentru a dezvolta o măsură de interdependenţă sau de coeziune [26].

A doua strategie poate fi adoptată prin introducerea unei măsuri de dependenţă pe baza energiei informaţionale [2]. Această măsură defineşte o dependenţă unilaterală între două variabile aleatoare:

$$o(C,T) = E(C|T) - E(C)$$

şi are următoarele proprietăţi:

a) o nu este simetrică în raport cu argumentele sale;

b) $o(C,T) \geqslant 0$, egalitatea îndeplinindu-se dacă şi numai dacă C şi T sunt independente;

c) $o(C,T) \leqslant 1 - E(C)$, iar egalitatea are loc dacă şi numai dacă C este complet dependentă de T.

Atunci când avem două variabile aleatoare C şi T astfel încât C este complet dependentă de T şi T este complet dependentă de C, se spune că C şi T sunt mutual complet dependente. Dacă C şi T sunt două variabile aleatoare astfel încât $o(C,T) = o(T,C)$, se spune că C şi T sunt egal unilateral dependente. Valoarea $o(C,T)$ corespunde cantităţii de informaţie despre C pe care o înglobează T, putând fi privită ca un indicator al dependenţei unilaterale ce caracterizează variabila aleatoare C în raport cu variabila aleatoare T. Se poate constata, aşadar, că $o(C,T)$ şi $I(C,T)$ măsoară practic acelaşi fenomen [2].

6.7 Algoritmul Energy Relevance LVQ (ERLVQ)

Am arătat în secţiunea anterioară că există o similaritate între informaţia mutuală $I(Y,X) = H(Y) - H(Y|X)$ şi valoarea $o(Y,X) = E(Y|X) - E(Y)$,

unde $E(Y)$ este energia informaţională a variabilei aleatoare Y. Iniţial, $o(Y,X)$ a fost definită pentru două variabile aleatoare discrete. Ea poate fi însă extinsă pentru variabile aleatoare continue sau variabile aleatoare continue împreună cu variabile aleatoare discrete.

Maximizăm $o(Y,C)$ pentru a obţine valorile relevanţelor:

$$\lambda^{(t+1)} = \lambda^{(t)} + \alpha \frac{\partial o(Y,C)}{\partial \lambda} \Leftrightarrow$$

$$\Leftrightarrow \lambda^{(t+1)} = \lambda^{(t)} + \alpha \sum_{i=1}^{N} \frac{\partial o(Y,C)}{\partial \mathbf{y}_i} \mathbf{I}\left(\mathbf{x}_i - \mathbf{w}_j\right). \tag{6.33}$$

Folosim următoarele notaţii: N - numărul total de eşantioane, M - numărul de clase, M_p - numărul de eşantioane din clasa c_p.

Conform definiţiei, putem scrie:

$$o(Y,C) = E(Y|C) - E(Y),$$

unde

$$\begin{aligned} E(Y|C) &= \int_{\mathbf{y}} \sum_{p=1}^{M} p(c_p) p^2(\mathbf{y}|c_p) d\mathbf{y} = \\ &= \sum_{p=1}^{M} p(c_p) \int_{\mathbf{y}} p^2(\mathbf{y}|c_p) d\mathbf{y} = \\ &= \sum_{p=1}^{M} \frac{1}{p(c_p)} \int_{\mathbf{y}} p^2(\mathbf{y}, c_p) d\mathbf{y} \end{aligned}$$

şi

$$E(Y) = \int_{\mathbf{y}} p^2(\mathbf{y}) d\mathbf{y}.$$

Obţinem

$$o(Y|C) = \sum_{p=1}^{M} \frac{1}{p(c_p)} \int_{\mathbf{y}} p^2(\mathbf{y}, c_p) d\mathbf{y} - \int_{\mathbf{y}} p^2(\mathbf{y}) d\mathbf{y}. \tag{6.34}$$

Calculul acestei expresii necesită un efort substanţial din punct de vedere computaţional, de aceea putem recurge la aproximarea cu ferestre Parzen a densităţilor pătratice de probabilitate care apar sub integrale. Folosim expresia gaussienei multidimensionale dată de relaţia (6.10), iar densitatea de probabilitate se poate aproxima prin (6.11).

Folosind aproximările cu ferestre Parzen avem:

$$\int_{\mathbf{y}} p^2(\mathbf{y}, c_p) d\mathbf{y} = \frac{1}{N^2} \sum_{k=1}^{M_p} \sum_{l=1}^{M_p} G(\mathbf{y}_{pk} - \mathbf{y}_{pl}, 2\sigma^2 \mathbf{I})$$

şi

$$\int_{\mathbf{y}} p^2(\mathbf{y})d\mathbf{y} = \frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N} G(\mathbf{y}_k - \mathbf{y}_l, 2\sigma^2\mathbf{I}).$$

Cu aceste relaţii, ecuaţia (6.34) se scrie:

$$\begin{aligned} o(Y,C) &= \frac{1}{N^2}\left(\sum_{p=1}^{M}\frac{1}{\frac{M_p}{N}}\right)\sum_{k=1}^{M_p}\sum_{l=1}^{M_p} G(\mathbf{y}_{pk} - \mathbf{y}_{pl}, 2\sigma^2\mathbf{I}) - \\ &\quad -\frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N} G(\mathbf{y}_k - \mathbf{y}_l, 2\sigma^2\mathbf{I}), \end{aligned}$$

adică

$$\begin{aligned} o(Y,C) &= \frac{1}{N}\left(\sum_{p=1}^{M}\frac{1}{M_p}\right)\sum_{k=1}^{M_p}\sum_{l=1}^{M_p} G(\mathbf{y}_{pk} - \mathbf{y}_{pl}, 2\sigma^2\mathbf{I}) - \\ &\quad -\frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N} G(\mathbf{y}_k - \mathbf{y}_l, 2\sigma^2\mathbf{I}). \end{aligned} \tag{6.35}$$

Pentru a dezvolta formula de actualizare a relevanţelor, folosim ideea din [83] conform căreia putem considera două eşantioane $\mathbf{y}_1$ şi $\mathbf{y}_2$ consecutive ca reprezentante ale claselor. Avem următoarele două cazuri:

a) Când cele două eşantioane aparţin aceleiaşi clase, $N = 2$, $M = 2$, $M_1 = 2$ şi $M_2 = 0$, deci primul termen al ecuaţiei (6.35) nu se poate calcula datorită nedeterminării introduse de valoarea lui M_2. Acest caz este, aşadar, ignorat.

b) Atunci când cele două eşantioane sunt din clase diferite, $N = 2$, $M = 2$, $M_1 = 1$ şi $M_2 = 1$. Ecuaţia (6.35) se poate scrie astfel:

$$\begin{aligned} o(Y,C) &= \frac{1}{2}\left(\sum_{p=1}^{2}\frac{1}{M_p}\right)\sum_{k=1}^{M_p}\sum_{l=1}^{M_p} G(\mathbf{y}_{pk} - \mathbf{y}_{pl}, 2\sigma^2\mathbf{I}) - \\ &\quad -\frac{1}{4}\sum_{k=1}^{2}\sum_{l=1}^{2} G(\mathbf{y}_k - \mathbf{y}_l, 2\sigma^2\mathbf{I}) = \\ &= \frac{1}{2}\left[G(\mathbf{y}_{11} - \mathbf{y}_{11}, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_{21} - \mathbf{y}_{21}, 2\sigma^2\mathbf{I})\right] - \\ &\quad -\frac{1}{4}\left[G(\mathbf{y}_1 - \mathbf{y}_1, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I}) + \right. \\ &\quad \left. + G(\mathbf{y}_2 - \mathbf{y}_1, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_2 - \mathbf{y}_2, 2\sigma^2\mathbf{I})\right] = \\ &= G(0, 2\sigma^2\mathbf{I}) - \frac{1}{2}\left[G(0, 2\sigma^2\mathbf{I}) + G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})\right]. \end{aligned}$$

În final relaţia devine:

$$o(Y,C) = G(0,2\sigma^2\mathbf{I}) - \frac{1}{2}G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I}). \tag{6.36}$$

Conform egalităţii (6.34), va trebui să calculăm derivatele parţiale ale lui $o(Y,C)$ în funcţie de $\mathbf{y}_1$ şi apoi de $\mathbf{y}_2$. Pentru aceasta, folosim relaţia următoare:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{y}_i}G(\mathbf{y}_i - \mathbf{y}_j, 2\sigma^2\mathbf{I}) &= G(\mathbf{y}_i - \mathbf{y}_j, 2\sigma^2\mathbf{I})\frac{\mathbf{y}_j - \mathbf{y}_i}{2\sigma^2} = \\ &= -\frac{\partial}{\partial \mathbf{y}_j}G(\mathbf{y}_i - \mathbf{y}_j, 2\sigma^2\mathbf{I}).\end{aligned}$$

Atunci:

$$\begin{aligned}\frac{\partial o(Y,C)}{\partial \mathbf{y}_1} &= \frac{\partial}{\partial \mathbf{y}_1}\left(G(0,2\sigma^2\mathbf{I}) - \frac{1}{2}G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})\right) = \\ &= \frac{\partial G(0,2\sigma^2\mathbf{I})}{\partial \mathbf{y}_1} - \frac{1}{2}\frac{\partial G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})}{\partial \mathbf{y}_1}.\end{aligned}$$

Deoarece

$$G(0,2\sigma^2\mathbf{I}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\sigma|^{\frac{1}{2}}}$$

este o constantă, avem:

$$\begin{aligned}\frac{\partial o(Y,C)}{\partial \mathbf{y}_1} &= -\frac{1}{2}\frac{\partial G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})}{\partial \mathbf{y}_1} = \\ &= -\frac{1}{2}G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I}) \cdot \frac{\mathbf{y}_2 - \mathbf{y}_1}{2\sigma^2}.\end{aligned}$$

Similar obţinem:

$$\begin{aligned}\frac{\partial o(Y,C)}{\partial \mathbf{y}_2} &= \frac{\partial}{\partial \mathbf{y}_2}\left(G(0,2\sigma^2\mathbf{I}) - \frac{1}{2}G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})\right) = \\ &= -\frac{1}{2}\frac{\partial G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I})}{\partial \mathbf{y}_2} = \\ &= \frac{1}{2}G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I}) \cdot \frac{\mathbf{y}_2 - \mathbf{y}_1}{2\sigma^2}.\end{aligned}$$

Înlocuind aceste rezultate în (6.33) rezultă:

$$\begin{aligned}\lambda^{(t+1)} &= \lambda^{(t)} + \alpha\left(-\frac{1}{2}G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I}) \cdot \frac{\mathbf{y}_2 - \mathbf{y}_1}{2\sigma^2}\mathbf{I}(\mathbf{x}_1 - \mathbf{w}_{j(1)}) + \right. \\ &\quad \left. + \frac{1}{2}G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I}) \cdot \frac{\mathbf{y}_2 - \mathbf{y}_1}{2\sigma^2}\mathbf{I}(\mathbf{x}_2 - \mathbf{w}_{j(2)})\right) \Rightarrow \\ \Rightarrow \lambda^{(t+1)} &= \lambda^{(t)} - \alpha\left(\frac{1}{2}G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I}) \cdot \frac{\mathbf{y}_2 - \mathbf{y}_1}{2\sigma^2}\mathbf{I}(\mathbf{x}_1 - w_{j(1)}) - \right. \\ &\quad \left. - \frac{1}{2}G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2\mathbf{I}) \cdot \frac{\mathbf{y}_2 - \mathbf{y}_1}{2\sigma^2}\mathbf{I}(\mathbf{x}_2 - \mathbf{w}_{j(2)})\right),\end{aligned}$$

sau

$$\lambda^{(t+1)} = \lambda^{(t)} - \alpha \frac{1}{4\sigma^2} G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2 \mathbf{I}) \cdot (\mathbf{y}_2 - \mathbf{y}_1) \mathbf{I}(\mathbf{x}_1 - \mathbf{w}_{j(1)} - \mathbf{x}_2 + \mathbf{w}_{j(2)}), \tag{6.37}$$

unde $\mathbf{w}_{j(1)}$ şi $\mathbf{w}_{j(2)}$ sunt prototipurile câştigătoare pentru vectorii de intrare $\mathbf{x}_1$ şi $\mathbf{x}_2$.

Următoarea procedură actualizează simultan prototipurile şi factorii de relevanţă:

Pasul 1. *Iniţializează η şi α. Iniţializează vectorul relevanţelor: $\lambda_k = \frac{1}{n}$, $k = 1, \ldots, n$.*

Pasul 2. *Iniţializează prototipurile.*

Pasul 3. *Actualizează vectorii prototip conform regulii LVQ modificate folosind distanţa D_{ij}:*

$$\mathbf{w}_j = \begin{cases} \mathbf{w}_j + \eta\lambda \mathbf{I}(\mathbf{x}_i - \mathbf{w}_j), & \text{dacă } \mathbf{x}_i \text{ a fost clasificat corect} \\ \mathbf{w}_j - \eta\lambda \mathbf{I}(\mathbf{x}_i - \mathbf{w}_j), & \text{în caz contrar} \end{cases}$$

Pasul 4. *Actualizează relevanţele:*

$$\begin{aligned} \lambda^{(t+1)} \quad = \quad & \lambda^{(t)} - \alpha \frac{1}{4\sigma^2} G(\mathbf{y}_1 - \mathbf{y}_2, 2\sigma^2 \mathbf{I}) \cdot (\mathbf{y}_2 - \mathbf{y}_1) \mathbf{I} \cdot \\ & \cdot (\mathbf{x}_1 - \mathbf{w}_{j(1)} - \mathbf{x}_2 + \mathbf{w}_{j(2)}). \end{aligned}$$

Pasul 5. *Transformă relevanţele:*

$$\lambda_k = \frac{e^{\lambda_k}}{\sum_{i=1}^{n} e^{\lambda_i}}$$

pentru $k = 1, \ldots, n$.

Pasul 6. *Calculează rangul fiecărei componente a vectorului de intrare.*

Pasul 7. *Repetă paşii 3-6 pentru fiecare vector de intrare.*

Remarcăm asemănarea algoritmilor MIRLVQ şi ERLVQ care, cu toate că sunt dezvoltaţi diferit, conduc la rezultate similare. Diferenţa este minoră şi apare la pasul 4, formula (6.37) putând fi înlocuită cu (6.32). Rezultatele experimentale sunt identice deoarece constanta care face diferentă între cele două variante introduce doar o scalare a ratei de instruire α.

6.8 Experimente

Am testat algoritmii MIRLVQ şi ERLVQ în aceleaşi condiţii ca şi OWA-RLVQ pentru a putea face o comparaţie obiectivă a rezultatelor şi a performanţelor lor. Deoarece MIRLVQ şi ERLVQ sunt similari, rezultatele înregistrate în tabelele 6.1, 6.2, 6.3 şi 6.4 sunt cele pentru ERLVQ. Ele sunt identice cu cele obţinute cu MIRLVQ, cu observaţia că în primul caz valoarea constantei α trebuie înjumătăţită. Importanţa componentelor poate fi diferită de cea obţinută pentru OWA-RLVQ din următorul motiv: OWA-RLVQ este focalizat pe o separare cât mai bună a centrelor clusterelor. ERLVQ încearcă, prin definiţie, o distincţie cât mai bună la limita dintre clustere deoarece foloseşte întotdeauna câte două eşantioane consecutive din clase diferite. Aşadar, OWA-RLVQ găseşte acele componente care sunt importante pentru accentuarea centrelor clusterelor, iar ERLVQ pe acelea care fac o discriminare cât mai bună între două clase adiacente. Aplicaţia C++ care implementează algoritmul ERLVQ este asemănătoare cu cea utilizată pentru testele din capitolul 5.

Am folosit mai întâi baza de date Iris [9] pentru care am instruit 6 prototipuri, identic cu testele pentru OWA-RLVQ, şi am obţinut o rată de recunoaştere de 97,33%, iar valorile parametrilor au fost $\eta = 0,15$ şi $\alpha = 20$. Pentru algoritmul MIRLVQ se foloseşte $\alpha = 40$. Rangurile componentelor vectoriale pe care le-am obţinut pentru această bază de date cu algoritmii RLVQ, OWA-RLVQ şi ERLVQ sunt în tabelul 6.1. Relevanţele componentelor vectoriale pentru ERLVQ sunt $[0{,}714\ 0{,}498\ 0{,}475\ 0{,}122]^t$. Rezultatele arată că atunci când se utilizează baza de date Iris, ultima componentă vectorială este cea mai importantă dacă ne propunem să construim clase ale căror centroizi sunt foarte accentuaţi, în timp ce dacă dorim să facem o discriminare cât mai bună între cele trei clase, prima componenta vectorială este cea mai semnificativă. Această concluzie este asemănătoare celor prezentate în [15] unde se arată că atunci când se folosesc doar primele două componente vectoriale se obţine o proiecţie în planul bidimensional a claselor "1" şi "2" cu centroizii relativ apropiaţi, dar cu clasele delimitate clar de o dreaptă, în timp ce atunci când se folosesc ultimele două componente, centroizii celor două clase proiectate în planul bidimensional sunt foarte depărtaţi, dar clasele se întrepătrund uşor, astfel că limita nu este atât de clară.

Pentru baza de date Deterding (Vowel Recognition) am păstrat acelaşi număr de 59 de prototipuri ca şi în testele anterioare. De această data, am obţinut o rată de recunoaştere de 47,18% faţă de 46,75% pentru OWA-RLVQ şi 46,32% pentru RLVQ. Parametrii de instruire au fost $\eta = 0,35$, $\alpha = 200$. Pentru algoritmul MIRLVQ se foloseşte $\alpha = 400$. În tabelul 6.2 avem rangul

Tabelul 6.1: Rangul componentelor vectoriale pentru baza de date Iris.

Rangul	1	2	3	4
Componentele RLVQ	4	2	3	1
Componentele OWA-RLVQ	4	3	2	1
Componentele ERLVQ	1	2	3	4

Tabelul 6.2: Rangul componentelor vectoriale pentru baza de date Deterding.

Rangul	1	2	3	4	5
Componentele RLVQ	2	5	1	9	6
Componentele OWA-RLVQ	8	2	4	5	6
Componentele ERLVQ	2	1	3	4	6
Rangul	6	7	8	9	10
Componentele RLVQ	3	4	8	7	10
Componentele OWA-RLVQ	9	3	7	1	10
Componentele ERLVQ	8	9	5	10	7

componentelor vectoriale pentru experimentele cu această bază de date şi componenta 2 este cea mai importantă atât pentru RLVQ cât şi pentru ERLVQ, în timp OWA-RLVQ a plasat-o pe locul 2. Componenta 7 are importanţa cea mai mică pentru algoritmul ERLVQ, iar RLVQ şi OWA-RLVQ o plasează pe penultimul, respectiv antepenultimul loc. Vectorul relevanţelor pe care l-am obţinut la testul prezentat aici este [0,354 0,439 0,349 0,310 0,271 0,299 0,253 0,287 0,284 0,268]^t.

În al treilea set de experimente am folosit baza de date Ionosphere [9] care este formată din vectori cu un număr mare de componente. Pentru a reprezenta cele două clase, am folosit 8 prototipuri aşa cum am făcut şi în experimentele cu OWA-RLVQ. Rata de recunoaştere pe care am obţinut-o a fost de 94,03%, spre deosebire de 93,37% cu OWA-RLVQ şi de 92,71% cu RLVQ. Valorile constantelor de instruire au fost $\eta = 0,45$, $\alpha = 0,9 \cdot 10^{13}$, iar pentru algoritmul MIRLVQ se foloseşte $\alpha = 1,8 \cdot 10^{13}$. Componenta vectorială cea mai importantă stabilită de acest algoritm este cea cu numărul 8. Componenta numărul 12 este plasată între primele 5 atât de ERLVQ cât şi de RLVQ şi de OWA-RLVQ, după cum se poate constata din tabelul 6.3.

Comparaţia ratelor de recunoaştere pentru algoritmii LVQ1, RLVQ, OWA-RLVQ şi ERLVQ din tabelul 6.4 arată că ultimul dintre ei oferă cele mai bune

Tabelul 6.3: Rangul componentelor vectoriale pentru baza de date Ionosphere. Sunt reprezentate cele mai importante 5 componente.

Rangul	1	2	3	4	5
Componentele RLVQ	20	28	26	12	6
Componentele OWA-RLVQ	14	12	1	3	28
Componentele ERLVQ	8	24	16	12	6

rezultate pentru toate cele trei baze de date testate. Pentru baza de date Deterding care reprezintă o cunoscută problemă de recunoaşterea vorbirii, creşterea de performanţă obţinută cu ERLVQ este de aproape 1%. Această metodă oferă rezultate satisfăcătoare şi pentru probleme generale de recunoaştere a formelor cum sunt Iris şi Ionosphere pentru care performanţele rămân superioare.

Tabelul 6.4: Comparaţie între ratele de recunoaştere obţinute cu LVQ1, RLVQ, OWA-RLVQ şi ERLVQ.

Baza de date	LVQ1	RLVQ	OWA-RLVQ	ERLVQ
Iris	91,33%	95,33%	96,66%	97,33%
Deterding	44,80%	46,32%	46,75%	47,18%
Ionosphere	90,06%	92,71%	93,37%	94,03%

6.9 Concluzii

Am introdus în acest capitol o nouă metodă de calcul al relevanţelor pentru componentele vectorilor de intrare într-un un sistem de recunoaştere a formelor. Cei doi algoritmi, MIRLVQ şi ERLVQ, sunt similari dar primul este dezvoltat prin maximizarea entropiei pătratice Renyi, iar al doilea maximizează o măsură bazată pe energia informaţională a lui Onicescu. MIRLVQ şi ERLVQ reprezintă o tehnică alternativă algoritmului OWA-RLVQ pentru calculul relevanţelor.

Algoritmii propuşi de noi oferă o altă perspectivă pentru recunoaşterea formelor, calculând relevanţele prin folosirea ca suport matematic al puternicului aparat reprezentat de teoria informaţiei. Metoda noastră stabileşte online valorile relevanţelor prin continua adaptare la fluxul de date de instruire. Relevanţele pot fi folosite la ordonarea în mod dinamic a componentelor vectorilor de intrare în funcţie de importanţa lor, dar şi la reducerea dimensiunii vectorilor prin renunţarea la componentele mai puţin importante.

Capitolul 7

Concluzii

7.1 Introducere

În acest capitol facem o sinteză a rezultatelor teoretice şi experimentale obţinute în lucrare. Secţiunea 7.2 prezintă contribuţiile pe care teza le aduce la dezvoltarea domeniului recunoaşterii formelor folosind reţele neurale, iar secţiunea 7.3 defineşte posibilele direcţii viitoare de dezvoltare a rezultatelor obţinute aici.

7.2 Contribuţii

Lucrarea abordează următoarele două probleme:

- Cum se poate transfera conceptul de concurenţă la nivelul unei colecţii de reţele neurale pornind de la ideea de modularitate?
- Care este impactul contribuţiei individuale a componentelor intrărilor asupra performanţelor unor sisteme de recunoaştere a formelor bazate pe reţele neurale?

Modelul reţelelor neurale concurente (CNN) din capitolul 4 este o soluţie consistentă pentru prima problemă, iar algoritmii OWA-RLVQ, MIRLVQ şi ERLVQ din capitolele 5 şi 6 reprezintă răspunsul nostru la a doua întrebare.

Contribuţiile pe care le aduce această lucrare se regăsesc în capitolele 3, 4, 5 şi 6.

În capitolul 2 am descris modelele neurale fundamentale şi principalele tehnici de instruire. Din punct de vedere istoric, neuronul McCulloch-Pitts este primul model neural care a fost descris în literatură şi pe care l-am prezentat în secţiunea 2.2. Acest neuron cu o structură foarte simplă are un mare potenţial de calcul, permiţând implementarea funcţiilor logice cu complexitate

oricât de mare. Perceptronul lui Rosenblatt prezentat în secţiunea 2.3 realizează un pas important înainte pentru că are capacitatea de a învăţa să separe vectori liniar separabili. Dacă perceptronii sunt organizaţi pe straturi, se poate realiza o structură complexă numită perceptron multistrat. Algoritmul de instruire *backpropagation of error* detaliat de noi în paragraful 2.7 este unul dintre cei mai cunoscuţi şi permite perceptronului multistrat să înveţe supervizat asocieri între domeniul intrărilor şi cel al ieşirilor chiar şi pentru clase neseparabile liniar. Reţelele neurale cu întârziere în timp descrise în secţiunea 2.8 sunt o extensie a perceptronului multistrat şi pot fi folosite pentru forme temporale. Aceste reţele neurale au fost introduse în special pentru aplicaţii de recunoaştere a vorbirii care, prin construcţia lor, înglobează factorul timp. Reţelele neurale cu autoorganizare pe care le-am analizat în secţiunea 2.9 au capacitatea de a extrage şi de a înmagazina trăsăturile esenţiale ale formelor de instruire şi, pe baza acestora, de a realiza clasificarea. Învăţarea competitivă este suportul unei clase de algoritmi care promovează selecţia celor mai performanţi neuroni ai unei reţele pentru care se aplică instruirea. Hărţile cu autoorganizare fac nesupervizat o proiecţie bidimensională a vectorilor de instruire. Algoritmii LVQ construiesc supervizat o mulţime de vectori care caracterizează întreg setul de instruire, fiind etichetaţi cu numele clasei pe care o reprezintă.

Recunoaşterea vorbirii cu modele clasice de reţele neurale

Este dificil să comparăm performanţele mai multor tipuri de reţele neurale atâta timp cât testele sunt realizate pe date diferite. Din acest motiv, nu putem trage o concluzie relevantă prin simpla alăturare a unor rezultate disponibile în diverse lucrări. Am preferat să facem un studiu comparativ al calităţii recunoaşterii cu perceptroni multistrat, reţele neurale cu întârziere în timp şi cu hărţi cu autoorganizare a unui set de cuvinte izolate, pe de o parte, şi a vorbitorilor dintr-un grup de subiecţi, pe de altă parte. Capitolul 3 detaliază metoda de lucru şi rezultatele pe care le-am obţinut. Am folosit bazele de date Speechdata şi Telephdata care conţin pronunţii în limba engleză ale tuturor cifrelor, prima dată înregistrate într-un studio special amenajat şi a doua oară înregistrate prin telefon. Datorită limitărilor pe care le produce transmisia pe liniile telefonice, a doua bază de date ridică dificultăţi suplimentare în recunoaştere. Metoda de preprocesare a semnalelor vocale descrisă în secţiunea 3.2 este binecunoscută. Aceasta constă din calculul coeficienţilor cepstrali pe ferestre de semnal care sunt suprapuse pentru a surprinde evoluţia temporală a semnalelor vocale. Experimental am stabilit că aceste modele neurale nu ne conduc la rezultate satisfăcătoare chiar dacă pentru testele de recunoaştere a

cuvintelor izolate avem rate de succes bune. Am putea argumenta că problemele de recunoaştere a vorbitorilor din Speechdata şi Telephdata pe care ne-am propus să le rezolvăm sunt dificile însă, cu toate acestea, rezultatele pe care le-am grupat în tabelul 3.2 din paragraful 3.3 nu sunt suficiente.

Reţele neurale concurente

Este evident că pe măsură ce o reţea neurală are de rezolvat o problemă din ce în ce mai puţin dificilă ea conduce la performanţe mai bune. Pentru a reduce complexitatea recunoaşterii printr-o singură reţea neurală a întregii colecţii de cuvinte izolate sau a tuturor vorbitorilor, am ales soluţia unei reţele neurale modulare formată din reţele neurale specializate pe subprobleme ale problemei iniţiale. În capitolul 4 am introdus noul model de clasificare al Reţelelor Neurale Concurente (*Concurrent Neural Networks* - CNN) ca o colecţie de reţele neurale de dimensiuni reduse care lucrează în paralel, iar clasificarea se face conform regulii *câştigătorul ia tot*.

Instruirea reţelelor neurale concurente se face conform schemei din figura 4.1 care presupune că fiecare modul este instruit cu propriul set de date. Sistemul este alcătuit din reţele neurale cu diverse arhitecturi. Noi am folosit module de tip perceptron multistrat, de tip reţea neurală cu întârziere în timp şi de tip hartă cu autoorganizare, însă se pot folosi şi alte variante. Schema de recunoaştere din figura 4.2 constă dintr-o colecţie de module instruite pe câte o subproblemă şi un modul care selectează cel mai bun răspuns. Algoritmii de instruire şi de recunoaştere din secţiunea 4.2 implementează aceste două tehnici care sunt particularizate în paragraful 4.3 pentru perceptroni multistrat, reţele neurale cu întârziere în timp şi hărţi cu autoorganizare. MLP-CNN şi TDNN-CNN folosesc module instruite supervizat şi seturile de vectori de instruire conţin atât exemple pozitive cât şi negative. Spre deosebire de acestea, SOM-CNN constă din module care sunt instruite printr-un algoritm nesupervizat şi datele constau doar din exemple pozitive. Secţiunea 4.3 oferă detaliile legate de testele de recunoaştere a vorbitorilor din bazele de date Speechdata şi Telephdata cu reţele neurale concurente. În comparaţie cu rezultatele din capitolul 3, de data aceasta progresele sunt substanţiale iar performanţele sunt, în medie, cu 30%-35% mai bune.

Remarcăm comportamentul foarte bun al SOM-CNN pe care îl folosim mai departe în recunoaşterea unor vocale din limba română. În acest scop, am realizat o bază de date care constă din secvenţe vocalizate extrase din pronunţii în limba română ale celor zece cifre. Am preprocesat aceste semnale vocale prin aceeaşi metodă folosită pentru Speechdata şi Telephdata pe care am descris-

o în paragraful 3.2. Energia semnalului calculată pe ferestre oferă o primă informaţie legată de prezenţa sau absenţa unei vocale şi ne permite să selectăm doar aceste zone. Identitatea vocalelor este stabilită apoi de o reţea neurală concurentă SOM-CNN.

Am realizat cinci tipuri de experimente diferite pentru a ne forma o imagine corectă asupra modelului propus de noi în acest capitol şi pentru o analiză detaliată a sa. Primul experiment a fost cel de recunoaştere a celor şase vocale şi semivocale cu care am instruit sistemul. Rezultatele experimentale au arătat în mod clar performanţele foarte bune ale SOM-CNN pentru care am obţinut un progres de peste 10% faţă de SOM, până la 84,21%. Celelalte experimente au analizat elemente foarte importante care contribuie la calitatea recunoaşterii: proporţia în care dimensiunile reale ale zonelor vocalizate şi nevocalizate sunt sesizate de SOM şi SOM-CNN, numărul de succesiuni vocală-semivocală recunoscute corect, proporţia în care sistemul sesizează prezenţa unei consoane în locul unei vocale sau a unei semivocale şi procentul în care sunt confundate vocalele. De fiecare dată când am folosit reţelele neurale concurente rezultatele au fost superioare. Putem considera că modelul propus de noi în capitolul 4 este o bună alternativă pentru sistemele clasice de recunoaştere a formelor.

Recunoaştere prin calculul relevanţelor intrărilor folosind operatori OWA

În capitolele 5 şi 6 analizăm problema impactului contribuţiei individuale a componentelor intrărilor asupra performanţelor unor sisteme de recunoaştere a formelor care folosesc reţele neurale.

Importanţele componentelor vectoriale ale intrărilor într-un sistem de recunoaştere a formelor, pe care le-am numit relevanţe sau factori de relevanţă, sunt calculate în capitolul 5 ca ponderi ale unor operatori speciali de agregare numiţi operatori OWA. Am prezentat această clasă de operatori în secţiunea 5.3, am subliniat proprietăţile lor matematice şi am prezentat o tehnică de instruire supervizată a lor. Metoda introdusă de noi prin algoritmul OWA-RLVQ foloseşte relevanţele pentru a defini un nou algoritm LVQ în care prototipurile sunt modificate după o regulă ce înlocuieşte distanţa euclidiană cu una care înglobează ponderile OWA. În paralel cu actualizarea prototipurilor sunt calculaţi şi factorii de relevanţă după o regulă descrisă în detaliu în paragraful 5.4. Această regulă se bazează pe ideea că un vector clasificat corect induce o mărire a relevanţelor, în timp ce unul clasificat incorect provoacă diminuarea acestor factori.

Am experimentat algoritmul OWA-RLVQ pe baze de date standard, dispo-

nibile în reţeaua Internet, pentru recunoaşterea formelor, după cum se poate vedea in paragraful 5.5. Am obţinut rezultate superioare atât faţă de algoritmul LVQ standard cât şi faţă de algoritmul RLVQ care calculează, de asemenea, relevanţele, însă după o metodă motivată euristic.

Algoritmul OWA-RLVQ este important deoarece face legătura între algoritmul RLVQ şi consistentul model matematic al operatorilor OWA. Metoda noastră are rezultate bune atât pentru probleme de recunoaştere vocală cât şi pentru probleme generale de recunoaştere a formelor. Relevanţele pot fi folosite şi pentru stabilirea dinamică a rangului fiecărei dimensiuni a vectorilor de intrare.

Recunoaştere prin calculul relevanţelor intrărilor folosind noţiuni de teoria informaţiei

În capitolul 6 introducem o nouă metodă de recunoaştere a formelor care foloseşte factori de relevanţă calculaţi prin maximizarea fie a informaţiei mutuale dintre intrări şi clase, fie a unei mărimi bazate pe energia informaţiei. Algoritmul ERLVQ, care se dovedeşte a fi similar cu MIRLVQ, este o alternativă la OWA-RLVQ deoarece calculează relevanţele care conduc la o bună delimitare a claselor, în timp ce al doilea algoritm calculează relevanţele care reorientează prototipurile claselor către centroizii acestora.

Informaţia mutuală se calculează ca o diferenţă a două entropii Shannon. Este cunoscută, însă, complexitatea de calcul pe care o implică estimarea entropiei Shannon prin eşantioane. Pentru a evita acest dezavantaj, am folosit o metodă recentă de aproximare a informaţiei mutuale prin estimarea densităţilor de probabilitate cu ferestre Parzen. Această metodă pe care am tratat-o în detaliu în secţiunea 6.3 foloseşte nuclee gaussiene multidimensionale. Una dintre proprietăţile lor remarcabile este că integrala produsului a două nuclee gaussiene este tot o gaussiană. În anexa B am demonstrat această proprietate. Estimarea densităţii de probabilitate cu ferestre Parzen se foloseşte pentru estimarea entropiei Renyi care include şi entropia Shannon ca un caz particular. Printr-o similaritate cu estimarea entropiei Renyi se poate estima informaţia mutuală sub forma informaţiei mutuale pătratice, după cum arătăm în paragraful 6.4.

Odată stabilit acest cadru formal, am dezvoltat algoritmul MIRLVQ printr-o metodă de gradient crescător pentru maximizarea informaţiei mutuale dintre variabila aleatoare continuă a intrărilor şi variabila aleatoare discretă a claselor. Pentru aceasta, mai întâi am introdus în secţiunea 6.5 algoritmul general *Weighted LVQ* care sintetizează etapele parcurse de un algoritm ce foloseşte relevanţele pentru clasificarea prin LVQ. OWA-RLVQ este o particularizare a

acestui algoritm, iar MIRLVQ se bazează tot pe acest şablon.

Elementul cheie al lui MIRLVQ este faptul că el actualizează relevanţele prin compararea apartenenţei la aceeaşi clasă a doi vectori de instruire consecutivi. Vectorii nu mai sunt apoi folosiţi, lucru care îi conferă acestui algoritm atributul de on-line. El poate fi folosit pentru fluxuri de date de instruire şi relevanţele se adaptează continuu, reflectând în mod dinamic schimbările care se petrec asupra intrărilor.

Algoritmul ERLVQ introdus în paragraful 6.7 este similar lui MIRLVQ, deşi este dezvoltat prin maximizarea unei mărimi bazate pe energia informaţională. Definiţia energiei informaţionale ne permite să folosim şi de această dată estimarea densităţilor de probabilitate cu ferestre Parzen şi obţinem o formulă de actualizare a relevanţelor pe baza unor vectori consecutivi din setul de instruire. Regulile de modificare iterativă a relevanţelor pe care le-am dedus prin cele două metode au fost integrate în algoritmul weighted LVQ rezultând procedura din secţiunea 6.7.

Am experimentat algoritmul ERLVQ în condiţii similare cu OWA-RLVQ pentru a putea face o comparaţie corectă a rezultatelor. Am constatat că abordarea prin prisma teoriei informaţiei conduce la rate de recunoaştere mai bune, inclusiv pentru baza de date Deterding care este un cunoscut benchmark în recunoaşterea vorbirii.

Cele două modele de recunoaştere a formelor sintetizate prin algoritmii OWA-RLVQ şi ERLVQ reflectă două abordări diferite ale aceleiaşi probleme. Factorii de relevanţă calculaţi fie prin prisma teoriei operatorilor OWA, fie prin cea a teoriei informaţiei conduc la rezultate care îmbunătăţesc ratele de recunoaştere în comparaţie cu algoritmi puternici care implementează aceeaşi gamă de aplicaţii. În plus, relevanţele sunt utile la ordonarea dinamică a componentelor vectorilor de intrare în funcţie de importanţa lor (*feature ranking*) sau la eliminarea componentelor puţin importante (*feature selection*).

Pentru orice tehnică de recunoaştere sau clasificare automată a formelor se poate face presupunerea că rezultatele obţinute ar putea fi dependente de datele de instruire şi de test. Acest lucru este adevărat în majoritatea cazurilor, motiv pentru care nu putem susţine superioritatea absolută a unei metode. Putem, însă, să folosim date cât mai diverse pentru a testa propriile tehnici sau ne putem verifica metodele în diferite contexte şi atunci spectrul cazurilor pentru care abordarea noastră este superioară altora se lărgeşte. Din această cauză am testat modelul CNN atât pentru recunoaşterea cuvintelor izolate cât şi pentru recunoaşterea vorbitorilor şi pentru recunoaşterea vocalelor. Dacă pentru primele două categorii de teste am folosit cuvinte în limba engleză, la

recunoaşterea vocalelor am utilizat cuvinte pronunţate în limba română. Atunci când am testat algoritmii OWA-RLVQ şi ERLVQ, pe lângă pronunţii vocale am folosit şi eşantioane care, în mod uzual, sunt folosite pentru sisteme generale de recunoaştere a formelor.

7.3 Direcţii viitoare de cercetare

În lucrare au fost discutate diverse variante de abordare a problemelor care au fost enunţate încă o dată la începutul acestui capitol. Există, desigur, şi alte idei care pot fi utile pentru extinderea acestor cercetări şi doresc să menţionez câteva dintre ele.

- Aşa cum au fost introduse aici, reţelele neurale concurente sunt colecţii de reţele neurale care sunt asociate câte unei singure clase de vectori. Este util de studiat cum poate fi modificat acest model în aşa fel încât să se optimizeze numărul de reţele neurale componente prin unificarea unor module fără a afecta performanţele generale. De această dată, vom avea module asociate mai multor clase. Această abordare *bottom-up* poate fi înlocuită cu una *top-down* prin care pornim cu o reţea neurală globală pe care o înlocuim cu mai multe reţele neurale concurente. Această variantă a CNN poate conduce la o modelare a creierului pe baza ipotezei că el constă din zone distincte care răspund la stimuli bine determinaţi;

- Pornind de la rezultatele cercetărilor din neurobiologie care au stabilit existenţa unei memorii de scurtă durată şi a unei memorii de lungă durată, se poate extinde modelul CNN prin transferarea unor vectori dintr-o clasă - memoria de scurtă durată - în altă clasă - memoria de lungă durată - atunci când se constată că ei generează erori de cuantizare minime;

- O continuare firească a rezultatelor materializate prin algoritmii OWA-RLVQ, MIRLVQ şi ERLVQ este folosirea relevanţelor pentru eliminarea dinamică a componentelor de intrare nesemnificative, cu utilitate deosebită în *data-mining*.

Majoritatea modelelor neurale abordează recunoaşterea formelor ca o problemă globală şi fără a face distincţia între aportul individual al intrărilor. Această teză demonstrează că tehnicile propuse de noi contribuie la sporirea acurateţii recunoaşterii şi creează suportul pentru cercetări viitoare legate de extinderea conceptului de concurenţă la nivelul sistemelor neurale şi de reducere a dimensiunii intrărilor prin eliminarea componentelor puţin importante.

Anexa A
Valori reprezentative ale coeficienţilor cepstrali pentru foneme din limba română

Tabelele B.1 şi B.2 conţin valorile numerice pe baza cărora a fost realizată figura 4.4. Aceste date sunt valori reprezentative ale coeficienţilor cepstrali calculate pentru câte o pronunţie a fiecărei vocale. Pronunţiile sunt ale primului dintre cei 4 vorbitori pe care i-am folosit la crearea bazei de date descrisă în secţiunea 4.4.2.

Tabelul B.1: Valori reprezentative ale coeficienţilor cepstrali calculate pentru câte o pronunţie a vocalelor A, E, I.

Coeficientul	A	E	I
0	8,54	7,07	5,66
1	-3,58	1,44	5,22
2	-1,33	1,51	3,11
3	0,55	-0,67	-1,91
4	1,82	-1,16	0,44
5	-1,04	0,31	-1,46
6	-0,27	0,06	-0,10
7	0,41	0,23	-0,58
8	5,71	4,68	5,01
9	-0,38	-0,10	-0,15
10	-0,10	-0,02	0,42
11	0,06	-0,18	0,04

Tabelul B.2: Valori reprezentative ale coeficienţilor cepstrali calculate pentru câte o pronunţie a vocalelor O, U, Ă.

Coeficientul	O	U	Ă
0	9,98	10,54	8,59
1	3,35	2,11	0,01
2	-0,40	-1,45	-2,68
3	-1,72	-1,48	-0,38
4	-0,67	-0,08	2,60
5	0,17	0,48	-0,63
6	-0,22	-0,22	-1,06
7	-0,15	-0,66	-0,40
8	4,88	5,41	4,38
9	0,18	-0,83	-0,48
10	0,36	-0,70	-0,31
11	0,17	-0,32	-0,31

Anexa B
Estimarea densităţilor de probabilitate cu ferestre Parzen. Demonstraţia proprietăţii (6.12)

Vom demonstra relaţia (6.12) conform căreia:

$$\int_{-\infty}^{+\infty} G(\mathbf{y}-\mathbf{y}_1, \sigma^2\mathbf{I})G(\mathbf{y}-\mathbf{y}_2, \sigma^2\mathbf{I})d\mathbf{y} = G(\mathbf{y}_1-\mathbf{y}_2, 2\sigma^2\mathbf{I}) \tag{B.1}$$

unde

$$G(\mathbf{y}, \sigma^2\mathbf{I}) = \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} e^{-\frac{\|\mathbf{y}\|^2}{2\sigma^2}}, \tag{B.2}$$

iar d este dimensiunea vectorilor $\mathbf{y}$, $\mathbf{y}_1$ şi $\mathbf{y}_2$.

Facem următoarele notaţii:

$$\mathbf{y} = [y_1, \ldots, y_d]^t$$

$$\mathbf{y}_1 = [y_{11}, \ldots, y_{1d}]^t$$

$$\mathbf{y}_2 = [y_{21}, \ldots, y_{2d}]^t.$$

Atunci, conform expresiei (B.2) obţinem:

$$G(\mathbf{y}-\mathbf{y}_1, \sigma^2\mathbf{I}) = \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} e^{-\frac{\|\mathbf{y}-\mathbf{y}_1\|^2}{2\sigma^2}} \tag{B.3}$$

şi

$$G(\mathbf{y}-\mathbf{y}_2, \sigma^2\mathbf{I}) = \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} e^{-\frac{\|\mathbf{y}-\mathbf{y}_2\|^2}{2\sigma^2}}. \tag{B.4}$$

Conform definiţiei normei scriem:

$$\|\mathbf{y}-\mathbf{y}_1\|^2 = (y_1-y_{11})^2 + \ldots + (y_d-y_{1d})^2 \tag{B.5}$$

şi

$$\|\mathbf{y}-\mathbf{y}_2\|^2=(y_1-y_{21})^2+\ldots+(y_d-y_{2d})^2. \tag{B.6}$$

Introducem (B.5) şi (B.6) în (B.3) şi (B.4) şi obţinem:

$$\begin{aligned} G(\mathbf{y}-\mathbf{y}_1,\sigma^2\mathbf{I}) &= \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}}e^{-\frac{(y_1-y_{11})^2+\ldots+(y_d-y_{1d})^2}{2\sigma^2}} \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}}e^{-\frac{(y_1-y_{11})^2}{2\sigma^2}}\cdot\ldots\cdot e^{-\frac{(y_d-y_{1d})^2}{2\sigma^2}} \end{aligned}$$

şi

$$\begin{aligned} G(\mathbf{y}-\mathbf{y}_2,\sigma^2\mathbf{I}) &= \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}}e^{-\frac{(y_2-y_{21})^2+\ldots+(y_d-y_{2d})^2}{2\sigma^2}} \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}}e^{-\frac{(y_1-y_{21})^2}{2\sigma^2}}\cdot\ldots\cdot e^{-\frac{(y_d-y_{2d})^2}{2\sigma^2}}. \end{aligned}$$

Deci:

$$\begin{aligned} &G(\mathbf{y}-\mathbf{y}_1,\sigma^2\mathbf{I})G(\mathbf{y}-\mathbf{y}_2,\sigma^2\mathbf{I}) = \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}}\cdot\frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}}e^{-\frac{(y_1-y_{11})^2+(y_1-y_{21})^2}{2\sigma^2}}\cdot\ldots\cdot e^{-\frac{(y_d-y_{1d})^2+(y_d-y_{2d})^2}{2\sigma^2}} \end{aligned}$$

de unde rezultă:

$$\begin{aligned} &\int_{-\infty}^{+\infty}G(\mathbf{y}-\mathbf{y}_1,\sigma^2\mathbf{I})G(\mathbf{y}-\mathbf{y}_2,\sigma^2\mathbf{I}) = \\ &= \int_{-\infty}^{+\infty}\frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}}\cdot\frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}}e^{-\frac{(y_1-y_{11})^2+(y_1-y_{21})^2}{2\sigma^2}}\cdot\ldots \\ &\ldots\cdot e^{-\frac{(y_d-y_{1d})^2+(y_d-y_{2d})^2}{2\sigma^2}}dx_1\ldots dx_d = \\ &= \frac{1}{(2\pi\sigma^2)^d}\left[\prod_{i=1}^{d}\int_{-\infty}^{+\infty}e^{-\frac{(y_i-y_{1i})^2+(y_i-y_{2i})^2}{2\sigma^2}}dy_i\right] = \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}}\left[\prod_{i=1}^{d}\frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{+\infty}e^{-\frac{(y_i-y_{1i})^2+(y_i-y_{2i})^2}{2\sigma^2}}dy_i\right]. \end{aligned} \tag{B.7}$$

Calculăm:

$$\begin{aligned} &\frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{+\infty}e^{-\frac{(y_i-y_{1i})^2+(y_i-y_{2i})^2}{2\sigma^2}}dy_i = \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{+\infty}e^{-\frac{(x-y_{1i})^2+(x-y_{2i})^2}{2\sigma^2}}dx = \\ &= \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{+\infty}e^{-\frac{1}{\sigma^2}\left[x^2-(y_{1j}+y_{2j})x+\left(\frac{y_{1j}+y_{2j}}{2}\right)^2-\left(\frac{y_{1j}+y_{2j}}{2}\right)^2+\frac{y_{1j}^2+y_{2j}^2}{2}\right]}dx = \\ &= \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{+\infty}e^{-\frac{1}{\sigma^2}\left[x-\frac{y_{ij}+y_{2j}}{2}\right]^2}\cdot e^{-\frac{1}{\sigma^2}\left[\frac{y_{1j}^2+y_{2j}^2}{2}-\frac{y_{1j}^2+y_{2j}^2+2y_{1j}y_{2j}}{4}\right]}dx = \end{aligned}$$

$$= e^{-\frac{1}{4\sigma^2}\left[y_{1j}^2+y_{2j}^2-2y_{1j}y_{2j}\right]} \cdot \frac{1}{\sqrt{2}} \cdot \frac{1}{\frac{\sigma}{\sqrt{2}}\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{-\frac{\left(x-\frac{y_{1j}+y_{2j}}{2}\right)^2}{2\left(\frac{\sigma}{\sqrt{2}}\right)^2}} dx =$$

$$= \frac{1}{\sqrt{2}} e^{-\frac{\left(y_{1j}+y_{2j}\right)^2}{4\sigma^2}},$$

unde am trecut de la variabila y_i la x pentru o scriere mai clară. În final, am folosit un rezultat care este cunoscut:

$$\frac{1}{\frac{\sigma}{\sqrt{2}}\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{-\frac{\left(x-\frac{y_{1j}+y_{2j}}{2}\right)^2}{2\left(\frac{\sigma}{\sqrt{2}}\right)^2}} dx = 1.$$

În concluzie, (B.7) devine:

$$\int_{-\infty}^{+\infty} G(\mathbf{y}-\mathbf{y}_1, \sigma^2\mathbf{I}) G(\mathbf{y}-\mathbf{y}_2, \sigma^2\mathbf{I}) d\mathbf{y} =$$

$$= \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \left(\prod_{j=1}^{d} \frac{1}{\sqrt{2}} e^{-\frac{(y_{1j}-y_{2j})^2}{4\sigma^2}} \right) =$$

$$= \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} \frac{1}{2^{\frac{d}{2}}} \left[e^{-\frac{1}{4\sigma^2}\sum_{j=1}^{d}(y_{1j}-y_{2j})^2)} \right] =$$

$$= \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} e^{-\frac{1}{4\sigma^2}\|\mathbf{y}_1-\mathbf{y}_2\|^2} =$$

$$= \frac{1}{[2\pi(2\sigma^2)]^{\frac{d}{2}}} e^{-\frac{\|\mathbf{y}_1-\mathbf{y}_2\|^2}{2\cdot(2\sigma^2)}} =$$

$$= G(\mathbf{y}_1-\mathbf{y}_2, 2\sigma^2\mathbf{I}).$$

Am scris ultima egalitate ţinând cont de definiţia (B.2). Relaţia (B.1) este astfel demonstrată.

Bibliografie

[1] R. Anand, K. Mehrotra, C.K. Mohan, S. Ranka "Efficient classification for multiclass problems using modular neural networks". IEEE Transactions on Neural Networks 6, 117-124, 1995.

[2] R. Andonie, F. Petrescu "Interacting systems and informational energy". Foundations of control and engineering 11, no. 2, 53-59, 1986.

[3] M. Andrei, I. Ghiţă "Limba română". Editura Didactică şi Pedagogică, Bucureşti, 1983.

[4] J. Basak, R.K. De, S.K. Pal "Unsupervised feature selection using a neuro-fuzzy approach". Pattern Recognition Letters 19, 997-1006, 1998.

[5] R. Battiti "Using mutual information for selecting features in supervised neural net learning". IEEE Transactions on Neural Networks 5, 537-550, 1994.

[6] G. Beliakov "How to build aggregation operators from data". International Journal of Intelligent Systems 18, 903-923, 2003.

[7] Y. Bengio, R. Cardin, R. de Mori, E. Merlo "Programable execution of multilayered networks for automatic speech recognition". Communications of the ACM 32, 195-199, 1989.

[8] C.M. Bishop "Neural networks for pattern recognition". Oxford University Press, New York, 1995.

[9] C.L. Blake, C.J. Merz "UCI repository of machine learning databases". University of California, Irvine, Department of Information and Computer Sciences, http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.

[10] T. Bojer, B. Hammer, D. Schunk, K.T. von Toschanowitz "Relevance determination in learning vector quantization". In M. Verleysen (Ed.), Proceedings of the European Symposium on Artificial Neural Networks (ESANN 2001), D-Side publications, 271-276, 2001.

[11] A. Caţaron, V.-E. Neagoe "Concurrent neural networks for speaker recognition". Proceedings of the International Conference on Telecommunications (ICT 2001), Bucureşti, 2001.

[12] A. Caţaron "Vowel recognition using concurrent neural networks". Proceedings of the 8th International Conference on Optimization of Electrical and Electronic Equipments (OPTIM 2002), Braşov, 779-782, 2002.

[13] A. Caţaron, S. Mătase "Speaker recognition using multi-layer perceptron, time delay neural network, self-organizing map and concurrent neural networks". Proceedings of the 8th International Conference on Optimization of Electrical and Electronic Equipments (OPTIM 2002), Braşov, 775-778, 2002.

[14] A. Caţaron, R. Andonie "RLVQ determination using OWA operators". In M.H. Hamza (Ed.), Proceedings of the 3rd IASTED International Conference on Artificial Intelligence and Applications (AIA 2003), September 8-10, Benalmadena, Spain, ACTA Press, 434-438, 2003.

[15] R.K. De, N.R. Pal, S.K. Pal "Feature analysis: neural network and fuzzy set theoretic approaches". Patterns Recognition 10(30), 1579-1590, 1997.

[16] J.R. Deller, J.G. Proakis, J.H.L. Hansen "Discrete-time processing of speech signals". Prentice Hall, Upper Saddle River, New Jersey, 1987.

[17] R.O. Duda, P.E. Hart, D.G. Stork "Pattern classification". John Wiley & Sons Inc., New York, 2001.

[18] J.L. Elman "Finding structure in time". Cognitive Science 14, 179-211, 1990.

[19] D. Erdogmus, J.C. Principe "Generalized information potential criterion for adaptive system training". IEEE Transactions on Neural Networks 13, 1035-1044, 2002.

[20] D. Filev, R.R. Yager "Learning OWA operator weights from data". Proceedings of the Third IEEE International Conference on Fuzzy Systems, Orlando, IEEE Press, 468-473, 1994.

[21] D. Filev, R.R. Yager "On the issue of obtaining OWA operator weights". Fuzzy Sets and Systems 94, 157-169, 1998.

[22] J.W. Fisher III, J.C. Principe "A methodology for information theoretic feature extraction". Proceedings of the IEEE World Congress on Computational Intelligence, Anchorage, Alaska, 1712-1716, 1998.

[23] B. Freisleben, C.A. Bohn "Speaker-independent word recognition with backpropagation networks". In R.F. Albrecht, C.R. Reeves, N.C. Steele (Eds.), Artificial neural nets and genetic algorithms, Proceedings of the International Conference in Innsbruck, Springer, Wien, 243-248, 1993.

[24] L.M. Fu "Neural networks in computer intelligence". McGraw-Hill, New York, 1994.

[25] R.M. Golden "Mathematical methods for neural network analysis and design". The MIT Press, Cambridge, Massachusetts, 1996.

[26] S. Guiasu "Information theory with applications". McGraw-Hill, New York, 1977.

[27] B. Hammer, T. Villmann "Input pruning for neural gas architectures". In M. Verleysen (Ed.), Proceedings of the European Symposium on Artificial Neural Networks (ESANN 2001), D-Facto publications, 283-288, 2001.

[28] B. Hammer, T. Villmann "Batch-RLVQ". In M. Verleysen (Ed.), Proceedings of the European Symposium on Artificial Neural Networks (ESANN 2002), D-Side publications, 295-300, 2002.

[29] B. Hammer, T. Villmann "Generalized relevance learning vector quantization". Neural Networks 15, 1059-1068, 2002.

[30] S. Haykin "Neural networks - a comprehensive foundation". Macmillan College Publishing Company, New York, 1994.

[31] D.O. Hebb "The organization of behaviour: a neuropsychological theory". Wiley, New York, 1949.

[32] D. Huang, T. Chow "Searching optimal feature subset using mutual information". In M. Verleysen (Ed.), Proceedings of the European Symposium on Artificial Neural Networks (ESANN 2003), D-Side publications, 161-166, 2003.

[33] I. Iordan, V. Robu "Limba română contemporană". Editura Didactică şi Pedagogică, Bucureşti, 1978.

[34] R.A. Jacobs, M.I. Jordan, A.G. Barto "Task decomposition through competition in a modular conectionist architecture: the what and where vision tasks". Cognitive Science 15, 219-250, 1991.

[35] G. Jumarie "Relative information". Springer-Verlag, Berlin, 1990.

[36] N.B. Karayiannis "An axiomatic approach to soft learning vector quantization and clustering ". IEEE Transactions on Neural Networks 10, 1153-1165, 1999.

[37] N.B. Karayiannis "Soft learning vector quantization and clustering algorithms based on ordered weighted aggregation operators". IEEE Transactions on Neural Networks 11, 1093-1105, 2000.

[38] N.B. Karayiannis, M.M. Randolph-Gips "Soft learning vector quantization and clustering algorithms based on non-euclidian norms: multinorm algorithms". IEEE Transactions on Neural Networks 14, 89-102, 2003.

[39] S.V. Kartalopoulos "Understanding neural networks and fuzzy logic". IEEE Press, Piscataway, New Jersey, 1996.

[40] T. Kohonen "The 'neural' phonetic typewriter". IEEE Computer Magazine, March, 11-22, 1988.

[41] T. Kohonen "Self-organization and associative memory". Series in Information Sciences 61, Springer-Verlag, Berlin, 1989.

[42] T. Kohonen "The self-organizing map". Proceedings of the IEEE 78, 1464-1480, 1990.

[43] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen "SOM_PAK: The self-organizing map program package". Laboratory of Computer and Information Science, Helsinki University of Technology, Finland, http://www.cis.hut.fi/research/som_research/nnrc_programs.html, 1995.

[44] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, K. Torkkola "LVQ_PAK: The learning vector quantization program package". Laboratory of Computer and Information Science, Helsinki University of Technology, Finland, http://www.cis.hut.fi/research/som_research/nnrc_programs.html, 1995.

[45] T. Kohonen "Self-organizing maps". Springer-Verlag, 1997.

[46] T. Kohonen, S. Kaski, K. Lagus, J. Salojarvi, J. Honkela, V. Paatero, A. Saarela "Self organization of a massive document collection". IEEE Transactions on Neural Networks 11, 574-585, 2000.

[47] M.A. Kraaijveld, J. Mao, A.K. Jain "A nonlinear projection method based on Kohonen's topology preserving maps". IEEE Transactions on Neural Networks 6, 548-559, 1995.

[48] N. Kwak, C.-H. Choi "Improved mutual information feature selector for neural networks in supervised learning". Proceedings of the International Joint Conference on Neural Networks (IJCNN 1999), vol. 2, 1313-1381, 1999.

[49] N. Kwak, C.-H. Choi "Input feature selection for classification problems". IEEE Transactions on Neural Networks 13, 143-159, 2002.

[50] N. Kwak, C.-H. Choi "Input feature selection by mutual information based on Parzen windows". IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 1667-1671, 2002.

[51] I. Lapidot (Voitovetsky), H. Guterman, A. Cohen "Unsupervised speaker recognition based on competition between self-organizing maps". IEEE Transactions on Neural Networks 13, 877-887, 2002.

[52] M. Last, A. Kandel, O. Maimom "Informatic-theoretic algorithm for feature selection". Pattern Recognition Letters 22, 799-811, 2001.

[53] W. Li "Mutual information functions versus correlation functions". Journal of Statistical Physics 60, 823-837, 1990.

[54] D.J.C. MacKay "Information theory, inference, and learning algorithms". Cambridge University Press, 2003.

[55] W.S. McCulloch, W Pitts "A logical calculus of the ideas immanent in nervous activity". Bulletin of Mathematical Byophisics 5, 115-133, 1943.

[56] W.S. McCulloch, W. Pitts "How we know universals: the perception of auditory and visual forms". Bulletin of Mathematical Byophisics 9, 127-147, 1947.

[57] P. Mitra, C.A. Murthy, S.K. Pal "Unsupervised feature selection using feature similarity". IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 301-312, 2002.

[58] V.-E. Neagoe, O. Cula "A fuzzy connectionist approach to vowel recognition". In B. Reusch and D. Dascalu (Eds.), Real World Applications of Intelligent Technologies (part II), Institutul Naţional de Cercetare şi Dezvoltate în Microtehnologii, Bucureşti, 1998.

[59] V.-E Neagoe, O. Stănăşilă "Recunoaşterea formelor şi reţele neurale - algoritmi fundamentali". Editura Matrix Rom, Bucureşti, 1999.

[60] V. Neagoe "Concurrent self-organizing maps for automatic face recognition". Proceedings of the 29th International Conference of the Romanian Technical Military Academy, Bucharest, Romania, 35-40, 2001.

[61] V. Neagoe, A. Ropot "Concurrent self-organizing maps for classification of multispectral sattelite imagery". Proceedings of the 6th World Multiconference on Systemics, Cybernetics, and Informatics (SCI 2002), Orlando, Florida, 126-131, 2002.

[62] V. Neagoe, A. Ropot "Concurrent self-organizing maps for pattern classification". Proceedings of the First IEEE International Conference on Cognitive Informatics (ICCI 2002), Calgary, Alberta, Canada, 304-312, 2002.

[63] V. Neagoe, I. Iatan "Face recognition using a fuzzy-gaussian neural network". Proceedings of the First IEEE International Conference on Cognitive Informatics (ICCI 2002), Calgary, Alberta, Canada, 361-368, 2002.

[64] V. Neagoe, A. Ropot "Concurrent self-organizing maps - a powerful tool for face and speaker recognition". Proceedings of the International Conference COMMUNICATIONS 2002, Edited by the Technical Military Academy, Bucharest, Romania, 276-281, 2002.

[65] O. Onicescu "Energia informaţională". Studii şi Cercetări Matematice 18, 1419-1420, 1966.

[66] F.J. Owens, R. Andonie, G.H. Zheng, A. Cataron, S. Manciulea "A comparative study of the multy-layer perceptron, the multi-output layer perceptron, the time-delay neural network and the Kohonen self-organizing map in an automatic speech recognition task". Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems (EIS 1998), ICSC Academic Press, 624-629, Tenerife, Spain, February 11-13, 1998.

[67] S.K. Pal, R.K. De, J. Basak "Unsupervised feature evaluation: a neuro-fuzzy approach". IEEE Transactions on Neural Networks 11, 366-376, 2000.

[68] M. Pregenzer, D. Flotzinger, G. Pfurtscheller "Distinction sensitive learning vector quantization - a noise-insensitive classification method". Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 1994), Orlando, Florida, 2890-2894, 1994.

[69] M. Pregenzer, G. Pfurtscheller, D. Flotzinger "Automated feature selection with a distinction sensitive learning vector quantizer". Neurocomputing 11, 19-29, 1996.

[70] J.C. Principe, D. Xu, J.W. Fisher III "Information-theoretic learning". In Simon Haykin (Ed.), Unsupervised adaptive filtering, Wiley, New York, 2000.

[71] I. Purcaru "Informaţie şi corelaţie". Editura Ştiinţifică şi enciclopedică, Bucureşti, 1988.

[72] L. Rabiner, B.-H. Juang "Fundamentals of speech recognition". Prentice Hall, Englewood Cliffs, New Jersey, 1993

[73] A. Rosetti, A. Lăzăroiu "Introducere în fonetică". Editura Ştiinţifică şi Enciclopedică, Bucureşti, 1982.

[74] J.G. Rueckl, K.R. Cave, S.M. Kosslyn "Why are 'what' and 'where' processed by separate cortical visual systems? A computational investigation". Journal of Cognitive Neuroscience 1, 171-186, 1989.

[75] D.E. Rumelhart, J.L. McClelland (Eds.) "Parallel distributed processing: explorations in the microstructure of cognition". MIT Press, Cambridge, MA, 1986.

[76] A. Sato, K. Yamada "Generalized learning vector quantization". In "Advances in neural information processing systems", G. Tesauro, D. Touretzky, T. Leen (Eds.), vol. 7, MIT Press, 423-429, 1995.

[77] R. Setiono, H. Liu "Neural-network feature selector". IEEE Transactions on Neural Networks 8, 654-662, 1997.

[78] C. Shannon "A mathematical theory of communication". The Bell System Technical Journal 27, 379-423, 623-656, 1948.

[79] V. Ştefănescu "Aplicaţii ale energiei şi corelaţiei informaţionale". Editura Academiei, Bucureşti, 1979.

[80] K. Torkkola, W.M. Campbell "Mutual information in learning feature transformations". Proceedings of the International Conference of Machine Learning, Stanford, CA, USA, 1015-1022, 2000.

[81] K. Torkkola "Nonlinear feature transforms using maximum mutual information". Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2001), Washington DC, USA, 2756-2761, 2001.

[82] K. Torkkola "Learning discriminative feature transforms to low dimensions in low dimensions". In "Advances in neural information processing systems" 14 (NIPS 2001), Vancouver, BC, Canada, MIT Press, 2001.

[83] K. Torkkola "On feature extraction by mutual information maximization". Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2002), Orlando, Florida, 2002.

[84] K. Torkkola "Feature extraction by non-parametric mutual information maximization". Journal of Machine Learning Research 3, 1415-1438, 2003.

[85] V. Torra "Learning weights for weighted OWA operators". Proceedings of the IEEE International Conference on Industrial Electronics, Control, and Instrumentation (IECON 2000), Nagoya, Japan, 2000.

[86] V. Torra "Weighted OWA operators for synthesis of information". Proceedings of the 5th IEEE International Conference on Fuzzy Systems, New Orleans, USA, 996-971, 2000.

[87] L.P.J. Veelenturf "Analysis and applications of artificial neural networks". Prentice Hall, London, 1995.

[88] J. Vesanto, E. Alhoniemi "Clustering of the self-organizing map". IEEE Transactions on Neural Networks 11, 586-600, 2000.

[89] X.L. Xie, G. Beni "A validity measure for fuzzy clustering". IEEE Transactions on Pattern Analysis and Machine Intelligence 13, 841-847, 1991.

[90] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K.J. Lang "Phoneme recognition using time-delay neural networks". IEEE Transactions on Acoustics, Speech, and Signal Processing 37, 328-339, 1989.

[91] A. Waibel, H. Sawai, K. Shikano "Modularity and scaling in large phoneme neural networks". IEEE Transactions on Acoustics, Speech, and Signal Processing 37, 1989.

[92] H. Watanabe "A new view of the formal entropy as a measure of interdependence and its application to pattern recognition". Proceedings of the IEEE International Conference on System, Man, and Cybernetics, Nashville, USA, 2827-2832, 2000.

[93] R.R. Yager "On ordered weighted averaging aggregation operators in multicriteria decisionmaking". IEEE Transactions on System, Man, and Cybernetics 18, 188-190, 1988.

[94] R.R. Yager, D. Filev "Induced ordered weighted averaging operators". IEEE Transactions on System, Man, and Cybernetics 29, 141-150, 1999.

[95] R.R. Yager "Induced aggregation operators". Fuzzy Sets and Systems 137, 59-69, 2003.

[96] D.S. Yeung, X.Z. Wang "Improving performance of similarity-based clustering by feature weight learning". IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 556-561, 2002.

[97] J.M. Zurada "Introduction to artificial neural systems". West Publishing Company, St. Paul, Minessotta, 1992.

[98] *** "Stuttgart Neural Network Simulator". SNNS Group, Institute for Parallel and Distributed High-Performance Systems (IPVR), University of Stuttgart, Germany, http://www-ra.informatik.uni-tuebingen.de/SNNS/, 1995.

Apariţii în anul 2004

[99] R. Andonie, A. Caţaron "An informational energy LVQ approach for feature ranking". Proceedings of the European Symposium on Artificial Neural Networks (ESANN 2004), Bruges, Belgium, 2004.

[100] A. Caţaron, R. Andonie "Computing OWA weights as relevance factors". Proceedings of the 9th International Conference on Optimization of Electrical and Electronic Equipments (OPTIM 2004), Braşov, 2004.

[101] A. Caţaron, R. Andonie "Energy generalized LVQ with relevance factors". Proceedings of the International Joint Conference on Neural Networks (IJCNN 2004), Budapest, Hungary, 2004.