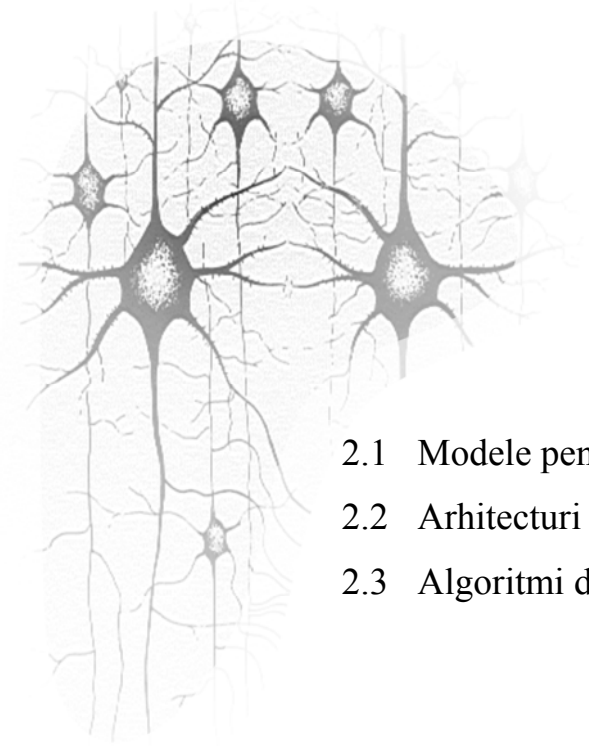
Capitolul **2** Caracteristici fundamentale ale rețelelor neurale artificiale



Rețelele neurale artificiale sunt caracterizate de 3 elemente: modelul adoptat pentru elementul de procesare individual (neuronul), structura particulară de interconexiuni (arhitectura) şi mecanismul de ajustare a legăturilor dintre neuroni (algoritmul de învățare). În cele ce urmează vom trece în revistă pe rând aceste elemente, prezentând exemple semnificative şi introducând terminologia corespunzătoare.

2.1 Modele pentru neuronul elementar

În prezentarea modelelor pentru neuronul elementar vom utiliza criteriile de clasificare folosite în mod uzual în teoria sistemelor, punct de vedere care va sugera metodele de analiză şi, în unele cazuri, de sinteză ale rețelelor studiate. În Fig. 2.1 se prezintă modalitatea uzuală de reprezentare grafică a neuronului individual, detaliată pentru varianta tipică a acestuia corespunzătoare aşa-numitului **model aditiv**.

A. După domeniul de definiție al semnalelor prelucrate:
a) modele analogice; b) modele discrete

Una dintre observațiile făcute în primul capitol sugera ideea potrivit căreia creierul este un "calculator" analogic. Fără a avea neapărat în prim plan criteriul plauzibilității biologice, dezbaterea referitoare la alegerea optimă între abordarea analogică sau discretă este un subiect de strictă actualitate. Argumentul cel mai puternic în favoarea primei alternative îl constituie viteza superioară recunoscută a calculului analogic, la care se adaugă lipsa necesității sincronizării (obligatorie în cazul rețelelor digitale cu funcționare sincronă şi care este, în general, dificil de asigurat în rețele de dimensiuni mari).

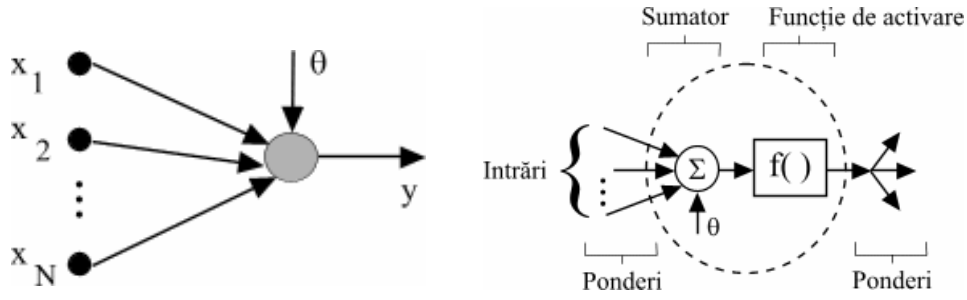


Fig. 2.1 : Modalitatea de reprezentare a neuronului elementar (model aditiv)

Avantajele abordării discrete rezidă în principal în precizia calculelor, importantă mai ales în cazurile în care parametrii rețelei sunt supuşi unor restricții severe, de exemplu referitoare la condiții de simetrie. Posibilitatea stocării pe durate mari de timp în formă nealterată a unor valori numerice utile reprezintă de asemenea un avantaj. Un aspect fundamental legat de implementarea rețelelor digitale îl constituie determinarea rezoluției necesare (a numărului de biți pe care se reprezintă valorile numerice) într-o aplicație dată.
O distincție suplimentară se poate face în raport cu gradul de cuantizare a semnalelor prelucrate. Se folosesc atât semnale necuantizate cât şi semnale cuantizate, de obicei binare[1]. Este important de subliniat că modelul discret nu presupune neapărat implementare digitală, ci poate fi folosită şi varianta care utilizează mărimi discrete necuantizate, folosind circuite cu capacități comutate.

B. După natura datelor prelucrate:
a) modele reale; b) modele complexe

În marea majoritate a cazurilor mărimile prelucrate sunt reale, dar în ultimul timp se utilizează şi rețele care lucrează cu variabile complexe sau, mai general, hipercomplexe[2]. Această alegere este justificată cu precădere în aplicații în care datele de intrare au o natură complexă intrinsecă (de exemplu, semnale radar sau unele semnale folosite în transmisiuni de date), precum şi de numărul mai redus de parametri necesari față de varianta reală. Algoritmii de învățare sunt, de regulă, extensii naturale ale variantelor formulate pentru semnale reale, însă atenție specială trebuie acordată în acest caz alegerii funcției de activare, în particular caracterului analitic al acesteia.

[1] Există exemple de rețele neurale care prelucrează semnale având mai multe nivele de cuantizare, care pot proveni din utilizarea unor funcții de activare multinivel [92] sau pot avea intrinsec un asemenea caracter, ca în cazul utilizării unor coduri multinivel (de exemplu, ternare) în transmisiuni de date.

[2] Numerele hipercomplexe generalizează noțiunea uzuală de număr complex. Un exemplu îl constituie *quaternionii* [71], care se pot scrie sub forma: $\mathbf{z} = z_0+z_1i+z_2j+z_3k$, unde i, j, k reprezintă cei trei vectori spațiali ortogonali, iar z_0-z_3 sunt parametri reali.

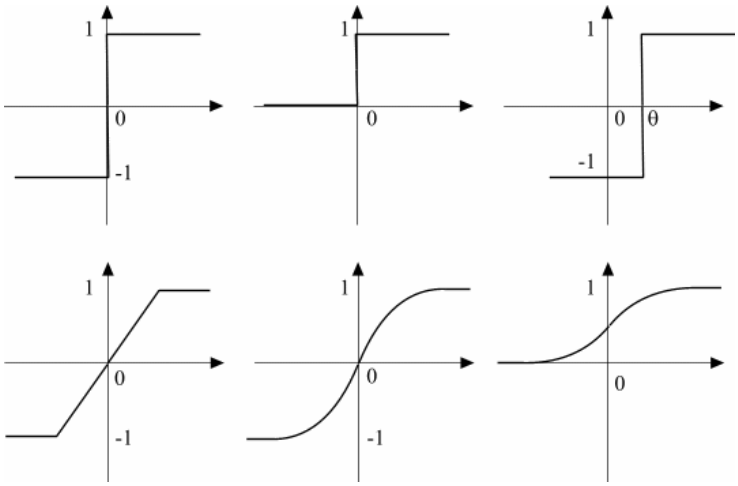


Fig. 2.2: Funcţii de activare pentru neuronul elementar:
a) comparator bipolar; b) comparator unipolar; c) comparator bipolar cu prag;
d) liniar cu saturaţie; e) sigmoidală bipolară; f) sigmoidală unipolară

C. După tipul funcţiei de activare:
a) modele liniare; b) modele neliniare

Funcţia de activare reprezintă funcţia de transfer intrare-ieşire a neuronului elementar. De departe, majoritatea reţelelor neurale artificiale întâlnite în literatură utilizează modele neliniare. Excepţia notabilă o constituie reţeaua de tip *Adaline* (prescurtare de la *ADAptive LInear NEuron*) şi varianta sa multidimensională *Madaline*, propuse de către profesorul american Bernard Widrow de la Universitatea Stanford [177]. Avantajul acestora îl constituie gama largă de algoritmi de învăţare performanţi existenţi, dar aria de aplicabilitate a reţelelor neurale care utilizează modele liniare este relativ restrânsă (egalizarea liniară a canalelor de transmisiuni de date, clasificatoare liniare). În Fig. 2.2 se prezintă câteva dintre funcţiile de activare des utilizate. Se pot face o serie de observaţii interesante:

• modelul de tip comparator (Fig. 2.2 a,b) poate fi întâlnit atât în reţele analogice cât şi în cele discrete

• modelul de tip comparator cu prag (Fig. 2.2 c) poate fi înlocuit cu un model fără prag dacă valoarea de prag θ se tratează ca o *intrare distinctă* de valoare constantă egală cu (-1) conectată printr-o legătură (pondere) care se va modifica în timp sub acţiunea algoritmului de învăţare

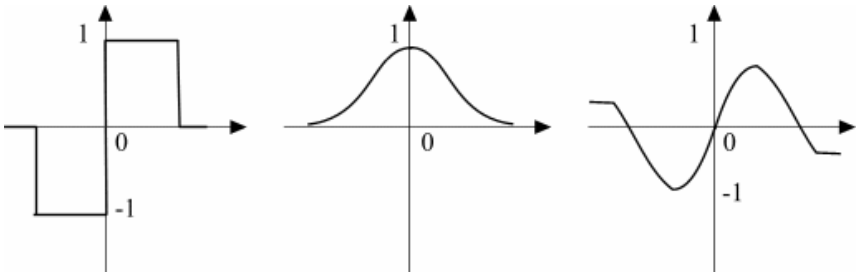


Fig. 2.3: Funcţii de activare nemonotone

• o justificare teoretică interesantă a performanţelor superioare pe care le asigură funcţiile de tip *sigmoidal* (Fig. 2.2 e,f) se prezintă în [70].
Deşi majoritatea funcţiilor de activare sunt monotone, există şi exemple de funcţii nemonotone care conduc la performanţe foarte bune (Fig. 2.3).

> *Observaţie:* Caracterul monoton (crescător) al funcţiei de activare constituie o cerinţă indispensabilă în formularea unor teoreme de convergenţă pentru o clasă largă de reţele neurale recurente [47], [83].

D. După prezenţa memoriei:
a) reţele cu memorie; b) reţele fără memorie

Memoria poate apare într-o reţea neurală pe 2 căi: datorită modelului adoptat pentru neuronii elementari şi, respectiv, datorită modelului adoptat pentru interconexiunile dintre aceştia. Primul caz este pus în evidenţă prin modalitatea de descriere a dinamicii individuale prin ecuaţii diferenţiale, respectiv, cu diferenţe sau chiar prin ecuaţii mixte, de tip diferenţial cu diferenţe. Al doilea caz este ilustrat de aşa-numitele reţele (discrete) cu sinapse dinamice, la care legăturile dintre neuroni nu sunt exprimate prin simple valori scalare, ci sunt reprezentate sub forma unor funcţii de transfer caracteristice filtrelor discrete cu răspuns finit sau infinit la impuls. O situaţie intermediară o constituie folosirea filtrelor *gamma* [150], care "împrumută" din avantajele ambelor tipuri. Un caz special îl constituie reţelele cu *memorie rezistivă*, obţinute prin considerarea unei funcţii de activare cu histerezis [95].
Reţelele fără memorie sunt reţele la care propagarea semnalelor se face numai dinspre intrare spre ieşire (*feedforward*), iar modelele adoptate atât pentru neuronii elementari cât şi pentru ponderi sunt strict algebrice. Aşa cum vom vedea în capitolele următoare,

pentru astfel de rețele există algoritmi de antrenare foarte puternici, de exemplu cei din categoria *backpropagation* (cu propagare inversă a erorii). Este important să subliniem că există exemple de rețele de tip *feedforward* cu reacție locală, utilizate mai ales în aplicații de prelucrare de semnale vocale. Unii autori identifică o aşa-numită memorie pe termen scurt (*short-time memory*), reprezentată de valorile variabilelor de stare ale sistemului şi o memorie pe termen lung (*long time memory*), dată de valorile interconexiunilor.
Din punctul de vedere al implementării, rețelele recurente ridică probleme speciale legate de necesitatea stocării unui volum mare de informații pe perioade însemnate de timp şi de elaborarea unor algoritmi de învățare suficient de rapizi pentru aplicații în timp real.

E. După dimensiunea spațiului stărilor pentru neuronul individual:
a) modele de ordinul I; b) modele de ordin superior

În cazul rețelelor *feedforward*, modelele considerate pentru neuronul elementar sunt de obicei de ordinul I şi se încadrează în aşa-numitul tip *aditiv,* potrivit căruia acesta efectuează o prelucrare în general neliniară asupra sumei ponderate a semnalelor aplicate la intrare (mărime care defineşte *activarea* neuronului):

$$y(x) = f\left(\sum_{i=1}^{N} w_i x_i\right) \tag{2. 1}$$

Au fost propuse şi modele de ordin superior, capabile să confere rețelelor formate din astfel de neuroni capacitatea de a surprinde corelații mai complexe ale datelor prelucrate, în particular posibilitatea de a asigura invarianța răspunsului rețelei la semnale de intrare obținute prin transformări elementare (translație, rotație) ale bazei de date originale. Exemplele cele mai cunoscute din acestă categorie sunt modelul *sigma-pi* [157] şi cel propus de către Giles şi Maxwell, bazat pe relația [66]:

$$y(x) = f\left(\sum_{i} w_i x_i + \sum_{i}\sum_{j} w_{ij} x_i x_j + \ldots\right) \tag{2. 2}$$

În ceea ce priveşte rețelele neurale recurente, analogice sau discrete, acestea sunt descrise de una din ecuațiile:

$$\dot{\mathbf{X}}(t) = F_W(\mathbf{X}(t), \mathbf{u}, \mathbf{X}(0)) \qquad (2.3)$$
$$\mathbf{X}[k+1] = F_W(\mathbf{X}[k], \mathbf{u}, \mathbf{X}[0]$$

unde matricea de interconexiuni **W** este determinată de aplicația concretă, vectorul **X** reuneşte variabilele de stare ale sistemului, **u** semnifică semnalul de intrare, iar **X**(0), **X**[0] desemnează condițiile inițiale. În cele mai multe modele prezentate în literatură neuronul individual este descris de o ecuație diferențială sau cu diferențe de ordinul I, exemplul tipic fiind oferit de rețeaua Hopfield [83]:

$$\dot{x}_i = -x_i + \sum_{j=1}^{N} w_{ij} f(x_j) \qquad (2.4)$$
$$x_i[n+1] = f\left(\sum_{j=1}^{N} w_{ij} x_j[n]\right)$$

unde N este numărul total de neuroni din sistem.
Recent au fost propuse şi modele de ordin superior pentru neuronii individuali, care utilizează în general oscilatoare pe post de elemente de procesare elementare. Un exemplu în acest sens îl reprezintă modelul de ordinul II introdus în [75]:

$$\dot{x}_i = -x_i + f\left(\sum_{j=1}^{N} w_{ij} x_j - K_i y_i + a u_i\right) \qquad (2.5)$$
$$\dot{y}_i = -y_i + f(K_i x_i)$$

unde $f(x) = (2/\pi)\tan^{-1}(x/a)$, iar a, K_i sunt constante reale. Mai mult, în [2] şi [5] se introduc modele de ordinul III, care prezintă particularitatea de a prezenta evoluție haotică chiar la nivelul unui neuron individual, cu efect favorabil în unele aplicații.
Deşi sunt mai bine motivate din punct de vedere biologic, modelele de ordin superior sunt mai dificil de analizat şi sintetizat la nivel de sistem, iar în unele aplicații rezultatele nu sunt mult mai bune față de varianta de ordinul I.

2.2 Arhitecturi specifice

Există numeroase modalităţi de interconectare a neuronilor elementari, care conduc la o evoluţie specifică a reţelei şi care se utilizează în aplicaţii dintre cele mai diverse. Pot fi identificate 2 clase distincte de arhitecturi:

• cu propagare a informaţiei numai dinspre intrare spre ieşire (reţele de tip *feedforward*). O particularitate constructivă a acestora o constituie posibilitatea de a identifica seturi de neuroni elementari grupaţi în aşa-numite "straturi", care oferă similitudini de conexiune. Ca terminologie, identificăm un strat de intrare, un strat de ieşire, iar toate celelalte sunt denumite straturi ascunse (*hidden layers*). Indexarea straturilor nu este tratată unitar în literatură (unii autori includ în numerotare şi stratul de intrare, alţii nu), însă de regulă este mai indicat să numărăm straturile de ponderi (interconexiuni). O variantă utilă în multe aplicaţii constă în separarea neuronilor din straturile ascunse în module distincte şi restricţionarea structurii de interconexiuni.

• reţele recurente (cu reacţie). Au fost introduse recent şi arhitecturi "mixte", al căror aspect global este *feedforward*, dar care prezintă reacţie locală. Este interesant de subliniat că semnalul de reacţie poate proveni de la stratul de ieşire, respectiv de la unul sau mai multe straturi ascunse.

Modalitatea de interconectare este diversă, mergând de la interconectarea neuronilor dintr-un strat numai spre stratul următor (în reţelele de tip f*eedforward* multistrat) până la reţele complet interconectate (recurente). Între aceste 2 extreme sunt cuprinse o multitudine de soluţii intermediare, dintre care enumerăm reţele *feedforward* generalizate, care permit şi conexiuni între neuroni aflaţi în straturi neînvecinate, reţele *feedforward* la care apar legături de reacţie între neuronii de pe acelaşi strat (reţele cu inhibiţie laterală) şi reţele la care legăturile de reacţie sunt prezente numai între neuronii elementari strict învecinaţi (reţele neurale celulare).

În Fig. 2.4 se indică arhitecturile cel mai des întâlnite, iar în Fig. 2.5 o serie de exemple mai “exotice”.

O clasă specială de circuite o constituie cele *local recurente*, la care reacţia este prezentă la nivelul modelului considerat pentru neuronii elementari, care sunt interconectaţi apoi în reţele *feedforward* obişnuite. Prezentăm în Fig. 2.6 schemele de principiu ale celor mai des utilizate.

În general, neuronii elementari sunt dispuşi într-un şir unidimensional în cadrul unui strat. Unele arhitecturi, de exemplu reţelele celulare [41] şi cele cu autoorganizare de tip Kohonen [103], pot avea straturi bidimensionale.

Din considerente legate de volumul de calcul necesar, dar şi ca urmare a existenţei unor rezultate teoretice riguroase, rareori se utilizează în practică reţele neurale cu mai mult de 3 straturi. Excepţii notabile sunt reţeaua de tip *counterpropagation* [79], precum şi unele variante de reţele autoasociative [49].

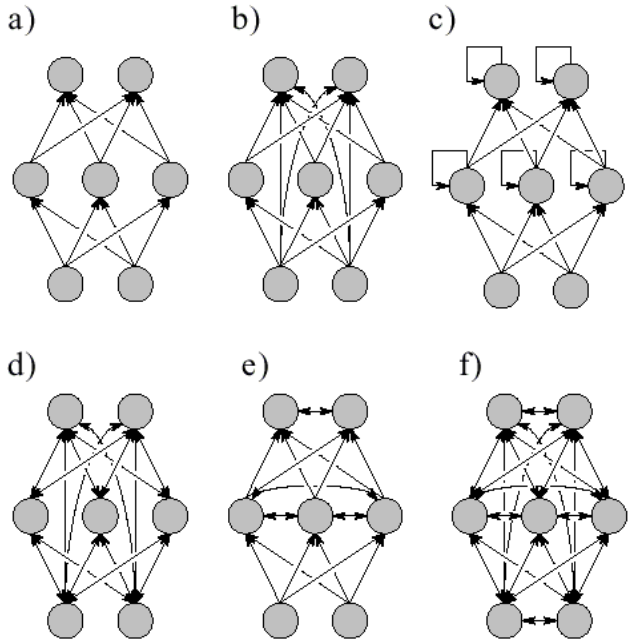
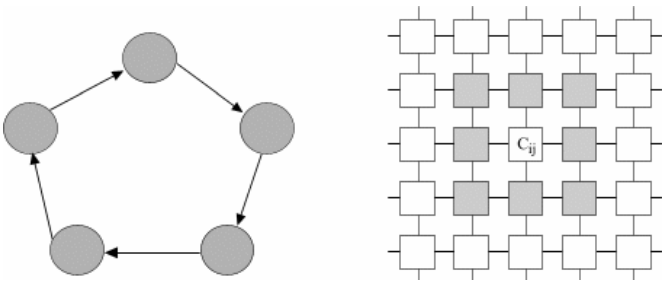


Fig. 2.4: Exemple de arhitecturi de rețele neurale artificiale:



a) b)

Fig. 2.5: a) Arhitectură de tip "inel"; b) rețea neurală celulară (CNN)

O problemă fundamentală o constituie modalitatea de a alege arhitectura adecvată pentru o aplicație dată. Lipsa unor teoreme constructive care să precizeze tipul rețelei şi numărul de neuroni elementari împreună cu modalitatea de interconectare dintre aceştia în vederea rezolvării unei anumite sarcini constituie în continuare una dintre principalele limitări ale utilizării rețelelor neurale artificiale şi totodată câmpul unor intense cercetări. Menționăm totuşi că există aplicații pentru care au fost formulate condiții minimale referitoare la arhitectură. Mai mult, în literatură se prezintă modalități de construcție sistematică urmând un proces iterativ, grupate în 2 categorii:

• tehnici de tip *pruning*, în care se pleacă de la sisteme de dimensiuni suficient de mari[3] şi se elimină pe rând neuronii elementari şi legăturile care se dovedesc neimportante (cele care nu se modifică semnificativ în procesul de învățare). Decizia de eliminare este de regulă bazată pe un calcul de senzitivitate al funcției de eroare în raport cu diversele ponderi ale sistemului. Un exemplu binecunoscut îl reprezintă metoda *Optimal Brain Damage* [109].

• tehnici de tip *learn and grow*, în care se pleacă de la rețele de dimensiuni reduse şi se adaugă neuroni şi conexiuni până când performanțele sistemului sunt suficient de bune. Ca exemple putem cita algoritmul *cascade-correlation* [57] şi metoda denumită *projection pursuit* [61].

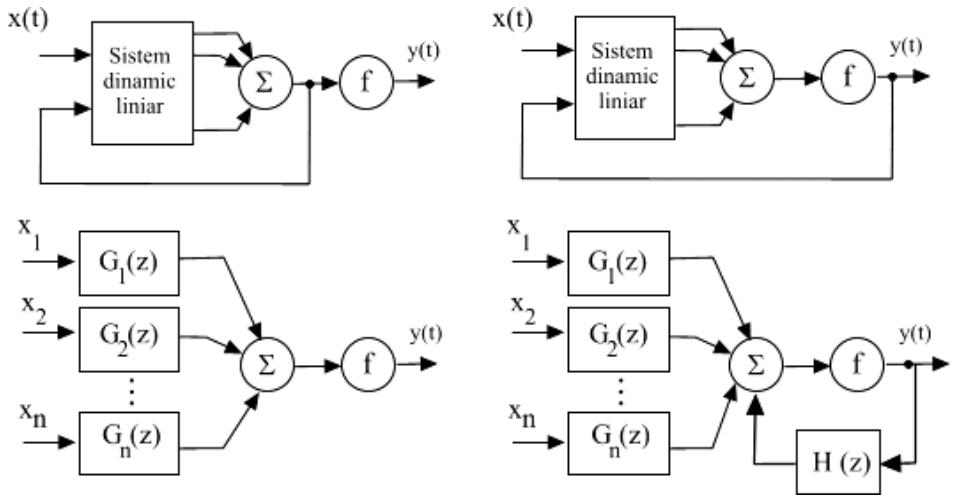


Fig. 2.6: Arhitecturi local recurente

[3] În realitate, este greu de apreciat ce înseamnă "suficient de mari".

2.3 Algoritmi de învățare

Unul dintre aspectele care diferențiază rețelele neurale față de alte sisteme de prelucrare a informației îl constituie capacitatea acestora de a învăța în urma interacțiunii cu mediul înconjurător şi, ca urmare, de a-şi îmbunătăți în timp performanțele (conform unui criteriu precizat). Deşi nu există o definiție general acceptată a procesului de învățare, prezentăm mai jos pe cea din [76]:

> *Învățarea este un proces prin care parametrii unei rețele neurale se adaptează în urma interacțiunii continue cu mediul de lucru. Tipul mecanismului de învățare este determinat de modalitatea concretă prin care se produce ajustarea valorilor parametrilor sistemului.*

Un aspect fundamental îl constituie modul de reprezentare internă a informațiilor care să permită interpretarea, predicția şi răspunsul corect la un stimul provenit din mediul înconjurător. O reprezentare corectă îi va permite sistemului, în particular rețelei neurale, să construiască un *model* al procesului analizat în stare să se comporte satisfăcător în condițiile în care la intrare i se vor aplica stimuli care nu au fost utilizați în procesul prealabil de învățare.
Informațiile utilizate în etapa de învățare (şi deci de sinteză a rețelei) sunt de 2 tipuri:
• informații disponibile *a priori* referitoare la particularitățile şi, eventual, restricțiile cărora le este supusă aplicația considerată. Astfel de considerente conduc, în general, la sisteme specializate de dimensiuni reduse, mai uşor de antrenat şi mai ieftine.
• informații sub forma unor perechi intrare-ieşire care surprind o relație de tip cauză-efect. Setul de date disponibil se împarte în două părți, una fiind folosită în procesul de modificare a ponderilor, deci de învățare propriu-zisă, iar cealaltă pentru a testa performanțele sistemului rezultat, oferind o imagine a aşa-numitei capacități de *generalizare* a rețelei.
Procesul de reprezentare internă respectă câteva reguli de bază, care sunt enumerate în continuare [8]:

Regula 1: Date de intrare similare trebuie să capete reprezentări interne asemănătoare.

Există mai multe moduri de a măsura "asemănarea" dintre 2 intrări distincte. Cea mai des folosită este cea bazată pe distanța Euclidiană dintre intrări (văzute ca vectori reali multidimensionali). Uneori se utilizează produsul scalar sau funcția de intercorelație dintre cele 2 mărimi.

Regula 2: Intrări aparținând unor categorii distincte trebuie să aibă reprezentări interne cât mai diferite.

Regula 3: În reprezentarea internă a unei caracteristici importante a datelor de intrare trebuie să fie implicați un număr mare de neuroni elementari.

Această regulă asigură un grad mare de încredere în luarea unei decizii şi toleranță sporită în cazul funcționării incorecte a unora dintre neuronii implicați în reprezentare.

Regula 4: Orice informație disponibilă *a priori*, precum şi eventuale invarianțe trebuie folosite în etapa de configurare (stabilirea arhitecturii şi a modului de interconectare) a rețelei.

Această indicație favorizează funcționarea corectă a rețelei deoarece aceasta nu trebuie să mai *învețe* particularitățile specifice aplicației considerate. Sistemele rezultate sunt în general *specializate*, având dimensiuni reduse, sunt mai uşor de implementat şi mai ieftine. Modalitățile de reprezentare a invarianțelor în raport cu diverse transformări elementare în rețele neurale se prezintă în [16].

2.3.1 Criterii de clasificare a algoritmilor de învățare

Există mai multe criterii în funcție de care se pot clasifica algoritmii de învățare, dintre care amintim:

A. În funcție de disponibilitatea răspunsului dorit la ieşirea rețelei neurale:
   a) învățare supravegheată; b) învățare nesupravegheată; c) învățare folosind un "critic"

• **învățarea supravegheată** (*supervised learning*) presupune existența *în orice moment* a unei valori dorite (*target*) a fiecărui neuron din stratul de ieşire al rețelei. Sistemului i se furnizează seturi de perechi intrare-ieşire dorită cu ajutorul cărora se calculează mărimi de eroare în funcție de diferența dintre valoarea reală a ieşirii şi cea dorită, pe baza cărora se ajustează valorile parametrilor rețelei (interconexiuni şi, eventual, valori de prag ale funcțiilor de activare). Exemple tipice: a) pentru rețele *feedforward:* algoritmul LMS (*Least Mean Square*) [176], clasa de algoritmi *back-propagation* (cu propagare inversă a erorii) [77], cuantizarea vectorială cu învățare

(LVQ) [103]; b) pentru reţele recurente: *backpropagation-through-time* [174], *real-time recurrent learning* [179].

• **în învăţarea nesupravegheată** (*unsupervised learning*) reţeaua extrage singură anumite caracteristici importante ale datelor de intrare formând reprezentări interne distincte ale acestora. Reţeaua nu beneficiază de seturi de ieşiri dorite, în schimb se utilizează un gen de "competiţie" între neuronii elementari care are ca efect modificarea conexiunilor aferente numai neuronului care "câştigă" întrecerea, restul legăturilor rămânând neafectate. Exemple din această categorie sunt: a) pentru reţele *feedforward: counterpropagation* [79]; b) pentru reţele recurente: algoritmul propus de Kohonen pentru hărţile cu autoorganizare (SOM) [103], algoritmul Hebb [78], Teoria Rezonanţei Adaptive (ART) elaborate de Grossberg [70]. În unele modele apare un parametru denumit intuitiv "conştiinţă" care intră în funcţiune atunci când unul dintre neuroni câştigă prea des competiţia.

• **învăţarea folosind un "critic"** (*reinforcement learning*) este denumită uneori şi cu recompensă/pedeapsă (*reward/punishment*). În această situaţie, reţeaua nu beneficiază de un semnal dorit, ca în cazul învăţării supravegheate, ci de un semnal care oferă o informaţie calitativă ilustrând cât de bine funcţionează sistemul (informaţia este binară, de tipul "răspunsul este bun/greşit", însă nu se indică şi cât de bun/greşit). Algoritmii aparţinând acestei categorii sunt inspiraţi într-o mai mare măsură de observaţii experimentale făcute pe animale şi, în esenţă, funcţionează după următorul principiu [76]: dacă urmarea unei anumite acţiuni întreprinse de un sistem capabil să înveţe are un efect favorabil, tendinţa de a produce acţiunea respectivă este încurajată, în caz contrar este inhibată.

În general algoritmii de învăţare respectă următoarea regulă [76]: vectorul multidimensional al ponderilor (interconexiunilor) aferente unui neuron elementar $\mathbf{W}_i$ se modifică proporţional cu produsul scalar dintre vectorul mărimilor de intrare $\mathbf{x}$ şi un aşa-numit "vector de învăţare" $\mathbf{r}$, reprezentat în general de o funcţie dependentă de $\mathbf{W}_i$, $\mathbf{x}$ şi, eventual, de vectorul ieşirilor dorite $\mathbf{d}$:

$$\mathbf{r} = \mathbf{r}(\mathbf{W}, \mathbf{x}, \mathbf{d}) \tag{2. 6}$$

Valoarea ponderilor se modifică după o relaţie de forma:

$$\Delta\mathbf{W} = \eta\, \mathbf{r}\, \mathbf{x} \tag{2. 7}$$

unde η este o constantă reală, de obicei subunitară, denumită constantă de învăţare.

B. În funcție de existența unui model analitic:
a) algoritmi parametrici; b) algoritmi neparametrici

Algoritmii parametrici presupun că procesul analizat poate fi modelat sub forma unei expresii matematice având o formă cunoscută, dependente de un număr (în general, restrâns) de parametri. Scopul urmărit în acest caz constă în estimarea cât mai exactă a valorilor acestor parametri pe baza datelor intrare-ieşire disponibile.
În cazul în care modelul considerat nu este adecvat, calitatea aproximării poate fi nesatisfăcătoare. În această situație sunt de preferat algoritmii neparametrici, care nu impun constrângeri de modelare. Astfel de algoritmi sunt capabili să aproximeze orice dependență intrare-ieşire, oricât de complicată, în virtutea unei aşa-numite capacități de aproximare universală pe care o posedă unii dintre aceştia.

C. În funcție de tipul aplicației pentru care sunt utilizați:
a) regresie; b) clasificare

Categoria cea mai răspândită de aplicații în care sunt utilizate rețelele neurale este cea de *aproximare funcțională*, în care se modelează dependențe dintre un set de variabile de intrare şi una sau mai multe variabile de ieşire. Setul de parametri care "traduc" această dependență este constituit din valorile interconexiunilor dintre neuroni, denumite de regulă *ponderi* sau *sinapse*. În modul cel mai general, o rețea neurală poate fi privită ca un mod particular de a stabili forma acestei dependențe, împreună cu modalitatea concretă de a fixa valorile parametrilor corespunzători folosind baza de date disponibilă.
Se pot distinge 2 categorii majore de aplicații: a) în clasificare se urmăreşte alocarea datelor aplicate la intrarea rețelei a uneia dintre "etichetele" corespunzătoare unui set *discret* de categorii avute la dispoziție (de exemplu, unei imagini reprezentând un caracter scris de mână i se asociază una dintre cele 26 de litere ale alfabetului). Din punct de vedere statistic, se urmăreşte de fapt aproximarea cât mai exactă a probabilității de apartenență a datelor de intrare la una dintre categoriile existente;
b) în cazul în care ieşirea rețelei poate avea valori *continue* avem de-a face cu o problemă de *regresie*, al cărei scop este aproximarea unei aşa-numite funcții de regresie (definită printr-o operație de mediere aritmetică a unei mărimi statistice specifice, ce va fi prezentată pe larg într-unul dintre paragrafele următoare). Regresia liniară este binecunoscută în analiza statistică, însă există aplicații practice importante (de exemplu, aplicațiile financiare) în care rezultatele obținute sunt nesatisfăcătoare, fiind necesară introducerea unui model neliniar.

În alegerea unui algoritm de învățare trebuie avute în vedere şi unele considerente de ordin practic, precum necesarul de memorie, viteza de convergență, complexitatea calculelor, comportarea în faza de testare. În Tabelul 2.1 se prezintă caracteristicile principalilor algoritmi de învățare, care vor fi tratați pe larg în capitolele următoare.

Tabelul 2.1: Principalele tipuri de algoritmi de învățare

Denumire	**Param. (P)/ Neparam. (N)**	**Clasificare (C)/ Regresie (R)**	**Caracteristici**	
			Necesar de memorie	**Viteză**
Regresie liniară	P	R	Foarte scăzut	Rapid
Backpropagation	N	R	Scăzut	Lent
Filtru Kalman	P	R	Ridicat	Rapid
LM	N	R	Mediu	Rapid
K-means	N	C	Mediu	Mediu
Projection pursuit	N	R	Scăzut	Mediu
SOM	N	C	Scăzut	Mediu
LVQ	N	C	Medu	Lent
Bayesian	N	C	Scăzut	Rapid

Legenda: LM – Levenberg-Marquardt; SOM – Self-Organizing Map; LVQ – Learning Vector Quantization

Să notăm în final cele 2 puncte de vedere distincte asupra procesului de învățare care se pot identifica analizând arhitecturile întâlnite în literatură, care explică în acelaşi timp şi capacitatea de generalizare a acestora:

• rețelele de tip *feedforward* tratează învățarea ca pe o **problemă de aproximare** a unei funcții de mai multe variabile (reale sau complexe) care exprimă legătura (necunoscută) dintre intrarea şi ieşirea sistemului pe baza unui set *finit* de exemple de tip intrare-ieşire dorită. O importanță deosebită o are dimensiunea bazei de date folosite în antrenare, aflată în strânsă dependență de numărul total de parametri ai rețelei (ponderi şi, eventual, valori de prag ale funcției de activare) şi care are un efect semnificativ asupra erorii de aproximare [19]. Necesitatea de a beneficia de o bază de date extrem de mare în cazul unor rețele cu mulți neuroni şi/sau straturi a fost denumită intuitiv "blestem al dimensionalității" (*curse of dimensionality*). De asemenea, este recunoscut pericolul de supraantrenare (*overfitting*), care constă în posibilitatea ca rețeaua să memoreze înseşi datele folosite în etapa de antrenare (în

general, însoţite de zgomot) şi nu să construiască un model al sistemului care le-a generat. În privinţa capacităţii reţelelor *feedforward* de a aproxima funcţii neliniare oarecare au fost elaborate analize teoretice extrem de riguroase, care pun în evidenţă proprietatea de aproximare universală a unor reţele cu funcţii de activare monotone de tip sigmoidal [52], [85] sau nemonotone de tip gaussian [139]. Au fost studiate şi condiţiile în care astfel de sisteme permit aproximarea simultană atât a funcţiei cât şi a derivatelor acesteia [86] şi au fost formulate estimări ale erorilor de aproximare. Reţelele de tip *feedforward* au fost utilizate cu succes în aplicaţii de clasificare, identificare de sistem, analiză a seriilor de timp.

• reţelele recurente codează informaţia sub forma **mulţimilor limită** ale unor sisteme dinamice neliniare multidimensionale [81]. Mulţimile limită (care, în mod intuitiv, reprezintă generalizarea noţiunii de regim permanent din cazul sistemelor liniare) conduc la unul dintre următoarele 4 tipuri de reprezentări în spaţiul stărilor: stări de echilibru stabil, cicluri limită corespunzătoare unor regimuri dinamice periodice, atractori specifici unor regimuri cvasiperiodice (de exemplu, cu aspect toroidal) şi atractori stranii, care pun în evidenţă prezenţa regimului de funcţionare haotic.

În cele mai multe situaţii se utilizează sisteme a căror dinamică evoluează spre puncte de echilibru stabil (sistemele sunt denumite *cu dinamică convergentă*), ale căror poziţii în spaţiul stărilor sunt fixate prin valorile interconexiunilor. În acest context, au fost raportate rezultate remarcabile în rezolvarea unor probleme de optimizare, de conversie analog-numerică şi de clasificare [84]. Analiza stabilităţii unor asemenea reţele se bazează de obicei pe metoda Liapunov [81], care prezintă avantajul de a nu necesita rezolvarea ecuaţiilor care descriu sistemul.

Recent se acordă un interes crescând şi reţelelor cu comportare periodică, în special în privinţa sincronizării ansamblurilor de oscilatoare elementare şi a stocării informaţiei sub forma ciclurilor limită. Mai mult, studiul reţelelor neurale cu comportare haotică este de asemenea avută în vedere, în special datorită raportării unor rezultate care confirmă existenţa unor astfel de regimuri în anumite zone ale creierului uman [62].

2.3.2 Funcţia de cost

Un aspect fundamental legat de procesul de învăţare al reţelelor neurale este cel referitor la scopul pentru care acestea sunt utilizate. Astfel, în cazul aplicaţiilor de regresie, se poate arăta că ţinta urmărită o constituie modelarea *densităţii de probabilitate* a valorilor de ieşire (*target*) condiţionată de distribuţia datelor de intrare. Pe de altă parte, în cazul problemelor de clasificare se urmăreşte estimarea probabilităţilor ca variabilele de intrare să aparţină uneia dintre categoriile disponibile. Atingerea acestor obiective devine posibilă prin optimizarea unor funcţii de cost

convenabil definite în funcție de parametrii rețelei neurale, cu observația că cele 2 tipuri de aplicații necesită de regulă folosirea unor funcții de cost specifice.
În cele ce urmează trecem în revistă o serie de aspecte teoretice fundamentale care vor permite înțelegerea mai exactă a modului de operare al rețelelor neurale.

A. Estimarea densității de probabilitate

În Anexa A sunt prezentate o serie de definiții ale unor noțiuni de bază din teoria probabilităților. Ne vom ocupa în cele ce urmează de posibilitatea de a modela o funcție de densitate de probabilitate p(**X**) folosind un număr **finit** de exemple **X**[n], cu n = 1,…N. Pornind de aici, vom ilustra în paragraful următor posibilitatea de a estima densități de probabilitate *condiționate*, care vor justifica în final scopul în care sunt folosite rețelele neurale.
Există 2 categorii de metode de estimare a densităților de probabilitate, anume metode parametrice, respectiv neparametrice. Cele dintâi impun o formă predefinită a funcției de densitate, dependentă de un număr de parametri specifici, ale căror valori urmează să fie estimate folosind baza de date disponibilă. Dezavantajul unei asemenea abordări constă în faptul că forma funcțională particulară impusă pur şi simplu se poate dovedi inadecvată modelării procesului fizic real care a generat datele. Metodele neparametrice nu particularizează forma funcției modelate, ci realizează estimarea pornind exclusiv de la datele disponibile, cu dezavantajul că numărul parametrilor necesari creşte pe măsură ce baza de date se lărgeşte. În cele ce urmează ne vom referi la o metodă parametrică de estimare bazată pe principiul denumit *maximum likelihood* [24]. Astfel, să considerăm o funcție densitate de probabilitate p(**X**) dependentă de un set de parametri $\boldsymbol{\theta} = [\theta_1\ \theta_2\ \ldots\ \theta_M]^T$ şi un număr de N vectori {**X**[1], **X**[2], …**X**[N]} care vor servi la estimarea acestor parametri. Densitatea de probabilitate a *ansamblului* acestor vectori (*joint probability density*) va fi:

$$L(\boldsymbol{\theta}) = \prod_{n=1}^{N} p_{\boldsymbol{\theta}}(\mathbf{X}[n]) \tag{2. 8}$$

care reprezintă o funcție ce depinde de variabilele $\boldsymbol{\theta}$ pentru un set *fixat* de vectori **X**[n]. Principiul denumit *maximum likelihood* urmăreşte determinarea valorilor vectorului de parametri $\boldsymbol{\theta}$ care asigură maximizarea funcției $L(\boldsymbol{\theta})$ (justificarea logică fiind legată de maximizarea probabilității ca datele disponibile să fi fost generate de către un model având parametrii optimi $\boldsymbol{\theta}$). Pentru ca procesul de optimizare să fie asociat cu noțiunea mai familiară a unei funcții de eroare – care ar trebui minimizată – se preferă înlocuirea funcției $L(\boldsymbol{\theta})$ prin versiunea sa procesată sub forma:

$$J = -\ln(L(\boldsymbol{\theta})) = -\sum_{n=1}^{N} \ln(p_{\theta}(\mathbf{X}[n])) \tag{2. 9}$$

Soluția acestei probleme de optimizare va depinde de forma particulară a funcției $p_{\boldsymbol{\theta}}(\mathbf{X})$ considerate şi, de regulă, va necesita utilizarea unui metode numerice adecvate. În cazul particular în care densitatea de probabilitate $p_{\boldsymbol{\theta}}(\mathbf{X})$ se presupune de formă *normală* (gaussiană) vom avea:

$$p(\mathbf{X}) = \frac{1}{(2\pi)^{d/2} \mid \boldsymbol{\Sigma} \mid^{1/2}} e^{-\frac{1}{2}(\mathbf{X}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{X}-\boldsymbol{\mu})} \tag{2. 10}$$

în care $\boldsymbol{\mu}$ reprezintă media aritmetică a vectorilor $\mathbf{X}$ (presupuşi de dimensiune d), iar $\boldsymbol{\Sigma}$ este matricea de covarianță, de dimensiune (dxd). Se poate arăta că procesul de minimizare a funcției J conduce în această situație la următoarele valori estimate ale parametrilor modelului (parametrii $\boldsymbol{\theta}$ se particularizează la valorile $\boldsymbol{\mu}$ şi $\boldsymbol{\Sigma}$**):**

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{X}[n] \tag{2. 11}$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{n=1}^{N} (\mathbf{X}[n]-\boldsymbol{\mu})(\mathbf{X}[n]-\boldsymbol{\mu})^T$$

Intuitiv, rezultatele obținute se justifică, ținând cont că înlocuind în relațiile anterioare operațiunea de mediere aritmetică pe un set finit de realizări individuale cu operatorul standard E{.} (*expectation*) ajungem la definițiile standard ale celor două mărimi valabile în cazul variabilelor cu distribuție normală.

B. Estimarea densității de probabilitate condiționată

Reamintim că scopul principal al unei rețele neurale este de a oferi un model cât mai exact al procesului fizic responsabil de generarea perechilor de date intrare-ieşire disponibile şi nu memorarea acestor valori particulare. Dacă scopul este atins, sistemul va furniza răspunsuri adecvate şi pentru date de intrare noi, care nu au fost utilizate efectiv în procesul de estimare a valorilor parametrilor specifici modelului. Instrumentul care permite descrierea procesului prin care sunt generate perechi de

vectori intrare-ieşire dorită este densitatea de probabilitate p(**X**,**t**), care se poate exprima în mod echivalent sub forma:

$$p(\mathbf{X},\mathbf{t}) = p(\mathbf{t} \mid \mathbf{X})p(\mathbf{X}) \tag{2. 12}$$

unde p(**t**|**X**) desemnează densitatea de probabilitate *condiționată* a ieşirii în raport cu intrarea (adică densitatea de probabilitate a variabilei **t** dacă intrarea **X** are o valoare particulară dată), iar p(**X**) este densitatea (necondiționată) de probabilitate a intrării. Ținând cont de definiția funcției L(**θ**) din paragraful anterior, se poate scrie în mod asemănător relația:

$$L = \prod_n p(\mathbf{X}[n],\mathbf{t}[n]) = \prod_n p(\mathbf{t}[n] \mid \mathbf{X}[n])p(\mathbf{X}[n]) \tag{2. 13}$$

Mai mult, trecând la varianta prelucrată sub forma unei funcții de eroare, se poate scrie:

$$J = -\ln(L) = -\sum_n \ln(p(\mathbf{t}[n] \mid \mathbf{X}[n])) - \sum_n \ln(p(\mathbf{X}[n])) \tag{2. 14}$$

Vom justifica imediat că scopul principal al unei rețele neurale va fi de a estima cât mai exact primul termen al relației anterioare. Deoarece cel de al doilea termen **nu depinde** de parametrii rețelei neurale putem renunța la acesta, funcția de eroare căpătând forma mai simplă:

$$J = -\sum_n \ln(p(\mathbf{t}[n] \mid \mathbf{X}[n])) \tag{2. 15}$$

Alegând diverse forme particulare ale densității de probabilitate condiționate p(**t**|**X**) se ajunge la definirea mai multor tipuri de funcții de eroare. Pentru simplitate, să considerăm în cele ce urmează că variabila aleatoare care defineşte semnalul dorit este unidimensională şi este obținută pe baza relației:

$$t[n] = h(\mathbf{X}[n]) + e[n], \quad n = 1 \ldots N \tag{2. 16}$$

în care h(.) desemnează o funcție deterministă, iar e[n] reprezintă zgomot cu distribuție normală (gaussiană) cu valoare medie nulă şi dispersie σ independentă de semnalul de intrare, de forma:

$$p(e) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{e^2}{2\sigma^2}} \tag{2. 17}$$

Să presupunem că avem la dispoziție o rețea neurală capabilă să ofere o aproximație a funcției h(.) sub forma $y_{\mathbf{W}}(\mathbf{X})$, în care vectorul **W** reuneşte totalitatea parametrilor rețelei. În aceste condiții relația (2. 3) se poate rescrie sub forma:

$$p(t \mid \mathbf{X}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t - y_{\mathbf{W}}(\mathbf{X}))^2}{2\sigma^2}} \tag{2. 18}$$

astfel încât funcția de eroare (2. 15) capătă forma:

$$J = \frac{1}{2\sigma^2} \sum_{n=1}^{N} \{t[n] - y_{\mathbf{W}}(\mathbf{X}[n])\}^2 + N \ln \sigma + \frac{N}{2} \ln(2\pi) \tag{2. 19}$$

Lăsând deoparte valorile constante (independente de valorile parametrilor **W**) ajungem în final la expresia binecunoscutei **erori pătratice**:

$$J = \frac{1}{2} \sum_{n=1}^{N} \{t[n] - y_{\mathbf{W}}(\mathbf{X}[n])\}^2 \tag{2. 20}$$

Observații: a) analiza anterioară poate fi extinsă comod la cazul variabilelor *target* multidimensionale

b) pentru precizarea completă a expresiei (2.18) este necesară şi obținerea valorii parametrului σ. Având la dispoziție valorile optime **W*** rezultate în urma minimizării funcției de eroare se poate demonstra că valoarea căutată este:

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^{N} \{t[n] - y_{\mathbf{W}^*}(\mathbf{X}[n])\}^2 \tag{2. 21}$$

c) nu este obligatoriu ca densitatea de probabilitate p(**t**|**X**) să aibă o distribuție normală, însă se poate arăta că în cazul utilizării funcției de eroare pătratice valorile optime ale parametrilor rețelei neurale nu vor putea face posibilă distincția între o distribuție normală şi oricare alt tip de distribuție având aceeaşi valoare medie şi aceeaşi dispersie.
d) în aplicațiile practice se folosesc deseori unele variante ale erorii pătratice (2. 20), anume:

Eroare pătratică medie (MSE):
$$J = \frac{1}{2N}\sum_{n=1}^{N}\left\{t[n] - y_{\mathbf{W}}(\mathbf{X}[n])\right\}^2 \qquad (2.\ 22)$$

Eroare pătratică medie normalizată (NMSE):
$$J = \frac{\sum_{n=1}^{N}\left\{t[n] - y_{\mathbf{W}}(\mathbf{X}[n])\right\}^2}{\sum_{n=1}^{N}\left\{t[n] - \mu_t)\right\}^2} \qquad (2.\ 23)$$

Varianta din relația (2. 22) prezintă avantajul independenței valorii erorii de numărul de exemplare care formează baza de date, iar cea din relația (2. 3) al unei imagini relative a valorii erorii în raport cu energia semnalului *target* (μ_t desemnează valoarea medie a datelor *target*).

C. Interpretarea ieşirilor unei rețele neurale

Având la dispoziție rezultatele foarte importante prezentate în paragrafele anterioare, vom ilustra în finalul acestui capitol modalitatea de interpretare a răspunsurilor oferite de ieşirile unei rețele neurale. Pentru simplitate, vom considera din nou cazul unei rețele cu o singură ieşire. Astfel, în cazul unei baze de date de dimensiune infinită (cu $N \to \infty$) expresia erorii pătratice (2. 20) devine [24]:

$$J = \lim_{N\to\infty} \frac{1}{2N} \sum_{n=1}^{N} \left\{ t[n] - y_{\mathbf{W}}(\mathbf{X}[n]) \right\}^2 = \\ = \frac{1}{2} \int \left\{ t[n] - y_{\mathbf{W}}(\mathbf{X}[n]) \right\}^2 p(t, \mathbf{X}) dt d\mathbf{X} \tag{2. 24}$$

Folosind relația (2. 12) putem scrie în continuare:

$$J = \frac{1}{2} \int \left\{ t[n] - y_{\mathbf{W}}(\mathbf{X}[n]) \right\}^2 p(t \mid \mathbf{X}) p(\mathbf{X}) dt d\mathbf{X} \tag{2. 25}$$

Introducem următoarele definiții ale unor medii aritmetice condiționate:

$$\langle t | X \rangle = \int t p(t \mid \mathbf{X}) dt \\ \left\langle t^2 \middle| X \right\rangle = \int t^2 p(t \mid \mathbf{X}) dt \tag{2. 26}$$

În urma unui calcul simplu se ajunge la următoarea expresie echivalentă cu relația (2. 25):

$$J = \frac{1}{2} \int \left\{ \langle t | \mathbf{X} \rangle - y_{\mathbf{W}}(\mathbf{X}) \right\}^2 p(\mathbf{X}) d\mathbf{X} + \frac{1}{2} \int \left\{ \left\langle t^2 \middle| \mathbf{X} \right\rangle - \langle t | \mathbf{X} \rangle^2 \right\} p(\mathbf{X}) d\mathbf{X} \tag{2. 27}$$

Se observă ușor că cel de al doilea termen din relația precedentă nu depinde de parametrii rețelei neurale, astfel încât minimizarea funcției de eroare va presupune anularea primului termen. Ajungem astfel la concluzia importantă că **atunci când se utilizează o funcție de eroare pătratică ieşirea unei rețele neurale poate fi interpretată ca valoarea medie a informației *target* condiționată de datele de intrare X**:

$$y_{\mathbf{W}}(\mathbf{X}) = \langle t | \mathbf{X} \rangle \tag{2. 28}$$

Interpretarea geometrică a acestei relații se indică în Fig. 2.7.

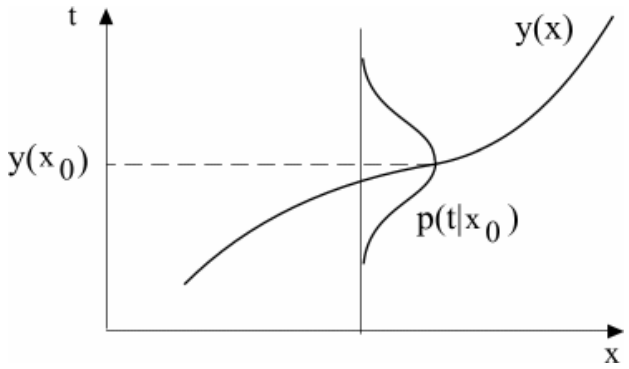


Fig. 2.7: Interpretarea ieşirii unei reţele neurale ca probabilitate condiţionată

Observaţii: a) analiza anterioară se poate extinde comod şi la cazul variabilelor *target* multidimensionale

b) concluziile fundamentale din ultimul paragraf pot fi aplicate în practică în condiţiile valabilităţii unor ipoteze care merită comentate. Prima se referă la necesitatea de a dispune de o bază de date de dimensiune foarte mare (pentru a permite trecerea de la sume finite la integrale). A doua se referă la capacitatea de modelare a sistemului care oferă ieşirea $y_{\mathbf{W}}(\mathbf{X})$ – şi care nu este obligatoriu să fie implementat sub forma unei reţele neurale! – în sensul că trebuie să avem garanţia că există un set de parametri **W** în stare să asigure minimizarea funcţiei de eroare. În sfârşit, presupunând că un astfel de set optim de parametri există, trebuie să avem la îndemână o tehnică de optimizare adecvată, capabilă să permită convergenţa către acest set de parametri. Din această perspectivă, utilizarea unor reţele neurale în vederea estimării densităţii de probabilitate (2. 3) se justifică prin *capacitatea de aproximare universală* pe care unele dintre acestea o posedă.

c) ieşirea $y_{\mathbf{W}}(\mathbf{X})$ poate fi furnizată atât de reţele neurale statice (strict algebrice), cât şi de reţele recurente. Pentru ca ieşirile unei reţele neurale să poată fi interpretate ca probabilităţi se folosesc de regulă funcţii de activare speciale pentru neuronii plasaţi în stratul de ieşire, pentru a asigura îndeplinirea celor 2 condiţii axiomatice referitoare la caracterul pozitiv, respectiv la condiţia ca suma acestora să fie egală cu 1 [138]. Un

exemplu în acest sens este oferit de funcția denumită *softmax*: $f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$, unde N este numărul total de ieşiri ale rețelei.

d) o utilizare extrem de utilă a noțiunilor prezentate în acest paragraf se întâlneşte în cazul aplicațiilor financiare de predicție, în care se preferă obținerea nu a unei valori punctuale ci estimarea întregii densități de probabilitate, cu efectul benefic al obținerii în acest mod a unei aprecieri a gradului de încredere în valoarea prezisă [134].