2. Perceptronul

Perceptronul este o rețea neuronală care realizează pe baza unui algoritm specific o clasificare a formelor de intrare (mărimi de intrare, date de intrare, vectori cheie etc.), respectiv o împărțire a acestora în anumite clase. Altfel spus intrările pentru instruire sunt organizate sub forma unei liste de asociere, în care fiecare element corespunde unei perechi formate dintr-un vector cheie (formă de intrare) – clasă de apartenență. Dacă au fost memorate suficient de multe obiecte din fiecare clasă, se poate construi o reprezentare internă "prototipică" a fiecărei clase prin ponderile de conexiune ale rețelei. În faza de lucru, rețeaua va stabili pentru o formă de intrare oarecare, clasa prototip corespunzătoare (aferentă).

Rețelele neuronale prezentate în această secțiune sunt adecvate pentru tratarea problemelor de clasificare. Intrările rețelei sunt vectorii ce urmează a fi clasificați iar ieşirile sunt clasele aferente. Algoritmii utilizați pentru instruire sunt de tipul de învățare supervizată.

Perceptronul reprezintă sâmburele din care s-au dezvoltat toate celelalte tipuri de rețele neuronale. Arhitectura perceptronului este cea mai simplă configurație posibilă a unei rețele neuronale, care se poate instrui cu un algoritm de asemenea simplu şi eficient. Acest algoritm este reprezentativ pentru o clasă largă de proceduri de instruire, motiv pentru care i se acordă o atenție deosebită

2.1. PERCEPTRONUL CU UN SINGUR STRAT

Configurația de bază a perceptronului cu un singur strat, similară cu structura unui neuron este prezentată în fig. 2. 1 , în care s-au folosit următoarele notații:

- $x_1, \ldots x_i, \ldots x_N$, reprezintă mărimile de intrare cu valori reale, $x_i \in \Re$,
- w_i sunt ponderile intrărilor cu $i = \overline{1,N}$
- y este ieşirea perceptronului, $\in \{-1,1\}$, sau $\{0,1\}$
- P - valoarea pragului de activare.

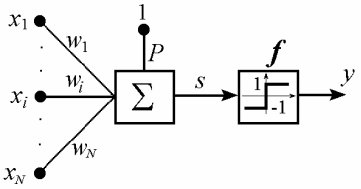

Fig. 2. 1 Perceptronul cu un singur strat

Funcția de activare a perceptronului este de tip signum.

$$f(s) = \begin{cases} 1 & \text{pentru } s + P \geq 0 \\ -1 & \text{pentru } s + P < 0 \end{cases}, \qquad (2.\ 1)$$

unde $s = \sum_{i=1}^{N} w_i x_i$ este activarea totală.

Din punctul de vedere al mărimilor de intrare se poate spune că perceptronul realizează o clasificare a acestora în două clase

- A_1 care corespunde ieşirii $y = 1$, respectiv
- A_2 care se obține pentru $y = -1$.

Pentru dezvoltarea algoritmului de instruire este indicat ca pragul de activare să fie introdus ca pondere pe o mărime de intrare suplimentară egală cu 1. Vectorii intrărilor şi al ponderilor vor fi în această situație:

$$\mathbf{x} = [x_0 \quad x_1 \ldots x_i \quad x_N]^T, \tag{2. 2}$$

respectiv

$$w = [w_0 \quad w_1 \ldots w_i \quad w_N]^T, \text{ cu } x_0 = 1 \text{şi } w_0 = P. \tag{2. 3}$$

Cu aceste notații ieşirea perceptronului se va calcula cu relațiile:

$$\mathbf{w}^T\mathbf{x} > 0 \quad \rightarrow \quad y = 1, \ \mathbf{w}^T\mathbf{x} < 0 \quad \rightarrow \ y = -1. \tag{2. 4), (2. 5}$$

Ecuația

$$\mathbf{w}^T\mathbf{x} = 0, \tag{2. 6}$$

reprezintă un hiperplan care împarte spațiul $\Re^{N+1}$ al vectorilor de intrare în două regiuni (clase). Vectorii de intrare aflați în partea pozitivă a hiperplanului de decizie aparțin clasei A_1 iar cei situați în partea negativă fac parte din clasa A_2. Hiperplanul de separare reprezintă o regiune de nedeterminare, deoarece vectorii care aparțin acestuia nu sunt incluşi în nici una dintre cele două clase.

În continuare în scopul simplificării algoritmului de instruire se va face o normalizare de semn a vectorilor de intrare, introducându-se variabila vectorială $\mathbf{z}$ definită conform relației:

$$\mathbf{z} = \begin{cases} \mathbf{x} & \text{dacă } \mathbf{x} \in A_1 \\ -\mathbf{x} & \text{dacă } \mathbf{x} \in A_2 \end{cases}. \tag{2. 7}$$

Cu această notație rezultă că hiperplanul $\mathbf{w}^T\mathbf{z} = 0$ este o regiune de separare a claselor A_1 şi A_2 dacă şi numai dacă există inegalitatea:

$$\mathbf{w}^T\mathbf{z} > 0 \ , \ \forall \mathbf{z} \in A_1 \cup A_2. \tag{2. 8}$$

Dacă acest hiperplan de separare există, cele două clase se vor numi liniar separabile. Ponderile conexiunilor şi pragul P (componentele vectorului $\mathbf{w}$) se vor ajusta folosind algoritmul de instruire al perceptronului, care este un algoritm de învățare supervizată (cu profesor, cu modele), care furnizează vectorul pondere ce separă cele două clase. Procedura de instruire a perceptronului constituie o metodă iterativă de optimizare a unei funcții criteriu. Pentru definirea funcției criteriu să considerăm un perceptron cu N intrări ($N+1$ dacă pragul este absorbit în mulțimea ponderilor).

Funcția criteriu

Admitem că mulțimea de instruire este formată din perechile:

$$(\mathbf{z}^1, d^1), \quad \ldots(\mathbf{z}^k, d^k), \quad \ldots(\mathbf{z}^p, d^p), \tag{2. 9}$$

unde $\mathbf{z}^k$ este vectorul normalizat de semn al mărimilor de intrare iar d^k este răspunsul corect (aşteptat, dorit) al rețelei. Notăm cu $E(\mathbf{w})$ mulțimea punctelor (vectorilor de intrare $\mathbf{z}$) eronat clasificate de către vectorul pondere $\mathbf{w}$ definită conform relației:

$$E(\mathbf{w}) = \{ \mathbf{z} \mid \mathbf{w}^{\mathrm{T}}\mathbf{z} < 0 \}. \qquad (2.\ 10)$$

Aşa cum am arătat ecuația $\mathbf{w}^{\mathrm{T}}\mathbf{z}$, unde $\mathbf{z}$ este o variabilă vectorială iar $\mathbf{w}$ un vector dat (cunoscut) reprezintă în spațiul $\Re^{N+1}$ un hiperplan, notat în continuare cu H. Pentru un punct oarecare $\mathbf{z}^k \in \Re^{N+1}$ care aparține mulțimii $E(\mathbf{w})$ eroarea de clasificare este distanța acestui punct la hiperplanul H, deci

$$e_k = d(\mathbf{z}^k, H). \qquad (2.\ 11)$$

Pentru evaluarea distanței de la un punct la un hiperplan din spațiul $(N+1)$ dimensional al vectorilor mărimilor de intrare să reamintim câteva noțiuni din geometria diferențială. Ecuația unui plan (hiperplan), H, care trece prin punctul $\mathbf{z}_0$, perpendicular pe direcția stabilită de vectorul $\mathbf{u}$ este:

$$\mathbf{u}^{\mathrm{T}}(\mathbf{z} - \mathbf{z}_0) = 0. \qquad (2.\ 12)$$

Distanța de la punctul $\mathbf{z}^k$ la acest hiperplan se calculează cu relația:

$$d(\mathbf{z}^k, H_1) = \frac{\left| \mathbf{u}^{\mathrm{T}}(\mathbf{z}^k - \mathbf{z}_0) \right|}{\|\mathbf{u}\|}, \qquad (2.\ 13)$$

unde $\|\mathbf{u}\|$ este norma euclidiană a vectorului normalei la hiperplanul H.

Deoarece hiperplanul H trece prin punctul $\mathbf{z}_0 = 0$ şi direcția considerată este dată de vectorul pondere $\mathbf{w}$, rezultă că eroarea de clasificare:

$$e_k = d(\mathbf{z}^k, H) = \frac{\left| \mathbf{w}^{\mathrm{T}}\mathbf{z}^k \right|}{\|\mathbf{w}\|}. \qquad (2.\ 14)$$

Având în vedere acum faptul că punctele eronat clasificate care aparțin mulțimii $E(\mathbf{w})$ îndeplinesc condiția $\mathbf{w}^{\mathrm{T}}\mathbf{z}^k < 0$ şi neglijând factorul de scalare $1/\|\mathbf{w}\|$ constant pentru un vector dat al ponderilor putem scrie în continuare eroarea de clasificare conform relației:

$$e_k = -\mathbf{w}^{\mathrm{T}}\mathbf{z}^k. \qquad (2.\ 15)$$

Funcția criteriu se obține prin însumarea erorilor de clasificare produse de un vector pondere pentru toate punctele eronat clasificate din $E(\mathbf{w})$, deci

$$J(\mathbf{w}) = -\sum_{\mathbf{z}^k \in E(\mathbf{w})} \mathbf{w}^{\mathrm{T}}\mathbf{z}^k,$$

sau în definitiv

$$J(\mathbf{w}) = -\sum_{\mathbf{z}\in E(\mathbf{w})} \mathbf{w}^{\mathsf{T}}\mathbf{z}\ ,\quad \mathrm{J}:\Re^{N+1}\to\Re\,. \tag{2. 16}$$

Algoritmul de instruire a perceptronului

Principiul algoritmului

Instruirea perceptronului se face pe baza unei metode de minimizare a funcției obiectiv J. Metoda de minimizare este de tip gradient, fiind denumită şi metoda celei mai mari pante, sau metoda celei mai abrupte coborâri. Conform acestei metode iterative vectorul pondere se determină cu relația

$$\mathbf{w}^{k+1} = \mathbf{w}^{k} - \alpha_k \mathbf{p}^{k}\,, \tag{2. 17}$$

unde $\alpha_k \in (0,1)$, iar

$$\mathbf{p}^{k} = \nabla J(\mathbf{w}^{k}) = \operatorname{grad} J(\mathbf{w})\big|_{\mathbf{w}=\mathbf{w}^{k}} = \left.\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}^{k}}\,. \tag{2. 18}$$

Pentru funcția criteriu definită mai înainte, având în vedere relația de derivare a produsului scalar

$$\frac{\partial}{\partial \mathbf{w}}(\mathbf{w}^{\mathsf{T}}\mathbf{z}) = \mathbf{z}\,, \tag{2. 19}$$

vom avea

$$\mathbf{p}^{k} = -\sum_{\mathbf{z}\in E_k}\mathbf{z}\,, \tag{2. 20}$$

deci

$$\mathbf{w}^{k+1} = \mathbf{w}^{k} + \alpha_k \sum_{\mathbf{z}\in E_k}\mathbf{z}\,, \tag{2. 21}$$

unde $E_k = E(\mathbf{w}^{k})$. Deoarece este dificil ca la fiecare pas să se determine mulțimea vectorilor de intrare clasificați eronat, pentru simplificarea algoritmului este mai convenabil ca, la fiecare pas, să se considere toate punctele (vectorii) din mulțimea de instruire. Practic, în loc de a calcula suma vectorilor $\mathbf{z}$, la fiecare pas k al algoritmului se consideră un singur vector de intrare $\mathbf{z}^{k}$. Vectorii de instruire $\mathbf{z}^{1}$, ..., $\mathbf{z}^{p}$ sunt incluşi într-un sistem de instruire de lungime suficient de mare folosind pentru n mai mare decât p proprietatea:

$$\mathbf{z}^{p+k} = \mathbf{z}^{\kappa}\,. \tag{2. 22}$$

În aceste condiții vectorul ponderilor $\mathbf{w}^{k+1}$ se modifică numai dacă $\mathbf{z}^{k}$ este eronat clasificat în caz contrar avem

$$\mathbf{w}^{k+1} = \mathbf{w}^{k}\,. \tag{2. 23}$$

De asemenea, în aplicarea corecției pe direcția antigradientului, pasul de corecție α_k se alege constant şi anume $\alpha_k = c$ unde $c \in (0,1)$ este numită creştere de corecție sau rată de învățare.

Algoritmul standard de instruire a perceptronului cu două clase pentru vectori normalizați
Ipoteză**:** vectorii de intrare sunt normalizați în raport cu semnul.

P₀ Se inițializează ponderile $w_1,.....,w_N$ şi pragul P obținându-se astfel vectorul $\mathbf{w}^1$. Ponderile inițiale se adoptă cu valori mici nenule. Se inițializează contorul numărului de paşi $k := 1$.

P₁ Se alege constanta de învățare $0 < c \leq 1$.

P₂ Se prezintă rețelei forma de intrare de instruire $\mathbf{z}^k$ şi ieşirea dorită d^k.

P₃ Se calculează ieşirea reală generată de perceptron, care este determinată de semnul expresiei $(\mathbf{w}^k)^T \mathbf{z}^k$

P₄ Condiția de oprire. Se repetă pasul **P₅** până când vectorul pondere nu se modifică un număr de p paşi consecutivi (p este numărul vectorilor de instruire).

P₅ Se adaptează ponderile şi pragul folosind relația:

$$\mathbf{w}^{k+1} = \begin{cases} \mathbf{w}^k + c\mathbf{z}^k & \text{dacă } (\mathbf{w}^k)^T \mathbf{z}^k \leq 0 \\ \mathbf{w}^k & \text{dacă } (\mathbf{w}^k)^T \mathbf{z}^k > 0 \end{cases}.$$

Se incrementează contorul $k := k + 1$

Observații:

1) Dacă algoritmul s-a oprit normal atunci vectorul $\mathbf{w}^{k+1}$ este o soluție a problemei de instuire.
2) În situația în care clasele A_1 şi A_2 nu sunt liniar separabile algoritmul perceptronului poate continua indefinit, şi procesul de stabilire a ponderilor nu se stabilizează. Se poate obține în acest caz o soluție aproximativă pentru vectorul pondere de separare modificând pasul **P₄** (condiția de oprire).

P₄ Condiția de oprire

Se repetă pasul **P₅** până când este satisfăcută una dintre condițiile:

i) Vectorul pondere nu s-a schimbat p paşi consecutivi.

ii) $k > N_0$, unde N_0 este numărul maxim admis de iterații

Nu există nici o metodă riguroasă sau euristică prin care să se poată previziona numărul N_0.

3) În literatură se demonstrează că algoritmul perceptronului este convergent într-un număr finit de paşi dacă vectorii de intrare aparțin unor clase liniar separabile.

Algoritmul standard al perceptronului pentru vectori nenormalizați
Normalizarea de semn a vectorilor de intrare s-a introdus pentru a facilita formularea şi rezolvarea matematică a problemei de optim care stă la baza algoritmului de instruire a perceptronului. În continuare se va reformula regula de corecție de la pasul **P₅** al

algoritmului prezentat mai înainte pentru cazul în care se utilizează vectori nenormalizați în raport cu semnul. Să considerăm acum că mulțimea de instruire este formată din perechile:

$$(\mathbf{x}^1, d^1), \quad \ldots(\mathbf{x}^k, d^k), \quad \ldots(\mathbf{x}^p, d^p), \tag{2. 24}$$

cu semnificațiile cunoscute. Vectorii $\mathbf{x}^k$ au $(N+1)$ componente din care prima cu valoarea 1. Ieşirea perceptronului la pasul k, notată y^k se calculează cu relația:

$$y^k = \begin{cases} 1 & \text{dacă } (\mathbf{w}^k)^{\mathrm{T}} \mathbf{x}^k \geq 0 \\ -1 & \text{dacă } (\mathbf{w}^k)^{\mathrm{T}} \mathbf{x}^k < 0 \end{cases}. \tag{2. 25}$$

Ieşirea corectă (dorită, aşteptată) asociată vectorului de intrare $\mathbf{x}^k$ este d^k unde

$$d^k = \begin{cases} 1 \text{ pentru } \mathbf{x}^k \in A_1 \text{ şi } \mathbf{z}^k = \mathbf{x}^k \\ -1 \text{ pentru } \mathbf{x}^k \in A_2 \text{ şi } \mathbf{z}^k = -\mathbf{x}^k \end{cases}. \tag{2. 26}$$

Regula de corecție a vectorului pondere se poate scrie acum condensat conform relației:

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \frac{c}{2}(d^k - y^k)\mathbf{x}^k. \tag{2. 27}$$

Analizând această relație se observă că dacă ieşirea dorită este $d^k = 1$ şi vectorul $\mathbf{x}^k$ care aparține clasei A_1 ($\mathbf{z}^k = \mathbf{x}^k$) este eronat clasificat, deci $y^k = -1$ atunci se produce o corecție

$$\mathbf{w}^{k+1} = \mathbf{w}^k + c\mathbf{z}^k = \mathbf{w}^k + c\mathbf{x}^k. \tag{2. 28}$$

Dacă vectorul este corect clasificat, ieşirea reală $y^k = 1$ şi vectorul pondere va fi

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \frac{c}{2}(1-1)\mathbf{x}^k = \mathbf{w}^k, \tag{2. 29}$$

rămânând aşadar neschimbat. Un raționament similar avem şi pentru vectorii $\mathbf{x}^k$ cu $d^k = -1$ ($\mathbf{z}^k = -\mathbf{x}^k$). În cazul unei clasificări incorecte $y^k = 1$ rezultă corecția:

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \frac{c}{2}(-1-1)\mathbf{x}^k = \mathbf{w}^k - c\mathbf{x}^k. \tag{2. 30}$$

Dacă vectorul este însă corect clasificat avem $y^k = -1$ şi vectorul pondere la pasul $(k+1)$ va fi egal cu cel de la pasul k.

Limitele perceptronului cu un singur strat

În multe din problemele concrete de clasificare intervin clase de obiecte care nu sunt liniar separabile. În acest caz perceptronul standard nu mai poate fi instruit pentru a clasifica vectorii de intrare. Un exemplu foarte simplu în acest sens şi totodată celebru este problema calculării valorilor funcției logice SAU-EXCLUSIV (XOR). Aceste valori se determină după cum se cunoaşte pe baza tabelului de adevăr prezentat în fig. 2.2.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Fig. 2. 2 Tabelul de adevăr al funcției logice XOR

Pentru rezolvarea problemei să considerăm un perceptron cu două intrări, x_1 şi x_2, cu ponderile w_1 şi w_2, având funcția de activare:

$$y = \begin{cases} 1 & \text{pentru } f(s) + P \geq 0 \\ 0 & \text{pentru } f(s) + P < 0 \end{cases}$$

unde P este valoarea pragului (fig. 2. 3). Problema de instruire este de a selecta ponderile w_1, w_2 şi pragul P astfel încât ieşirile perceptronului să furnizeze valorile funcției XOR. Reprezentând în plan cele patru perechi binare care sunt vectorii de intrare ai perceptronului se poate constata uşor că nu există nici o dreaptă (echivalentul spațial cu două dimensiuni al hiperplanului) care să separe planul astfel încât punctele (0,1) şi (1,0), care produc aceeaşi ieşire, să fie într-un semiplan, iar punctele (0,0), (1,1) să aparțină celuilalt semiplan. Deci perceptronul standard nu poate rezolva problema XOR.

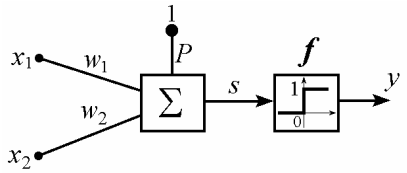

Fig. 2. 3 Modelul perceptronului cu două intrări

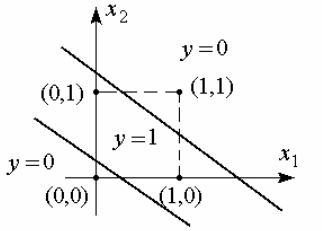

Fig. 2. 4 Regiunile de decizie pentru problema XOR.

Pentru clasificarea corectă a vectorilor de intrare în cazul problemei XOR, ar trebui ca planul variabilelor x_1, x_2 să fie separat ca în fig. 2. 4. Împărțirea planului în astfel de regiuni de decizie se poate obține cu o rețea neuronală (perceptron) cu două straturi cu arhitectura prezentată în fig. 2. 5

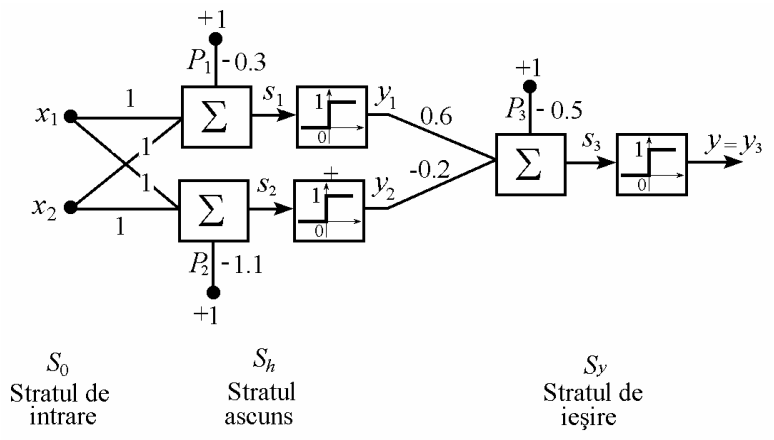

Fig.2. 5 Perceptronul cu două straturi.

Stratul de intrare notat S_0 este format din unități de calcul care nu realizează de fapt procesele de prelucrare specifice neuronilor. Practic, acest strat conține un număr de noduri (egal cu numărul intrărilor) de ramificare a intrărilor rețelei la neuronii din stratul ascuns. Elementele de calcul din acest câmp se mai numesc noduri sursă. Din acest motiv acest strat nu este luat în considerare la stabilirea numărului total de straturi ale arhitecturii rețelei şi rețeaua prezentată mai înainte se numeşte cu două straturi. Menționăm totuşi că în unele lucrări din literatură câmpul de intrare este luat în considerare la stabilirea numărului total de straturi. În cadrul lucrării convenim să ignorăm acest strat când se defineşte tipul arhitecturii unei rețele neuronale. În fig.2. 6 se prezintă sub formă tabelară valorile mărimilor de ieşire în funcție de cele patru combinații posibile ale variabilelor de intrare.

x_1	x_2	y_1	y_2	s_3	$y = y_3$
0	0	0	0	0	0
0	1	1	0	0,6	1
1	0	1	0	0,6	1
1	1	1	1	0,4	0

Fig. 2.6 Funcționarea perceptronului cu două straturi la rezolvarea problemei XOR.

Algoritmul buzunarului (pocket)

S. Gallant a propus în 1969 o modificare a algoritmului perceptronului, prin care acesta devine aplicabil chiar şi pentru unele clase neseparabile. Algoritmul pocket urmăreşte determinarea unui vector pondere, numit optimal, capabil să clasifice corect numărul maxim posibil de forme de instruire. Ideea de bază a algoritmului pocket constă în a păstra în “buzunarul nostru” vectorul pondere notat în continuare **b** care a realizat cea mai lungă secvență de clasificări consecutive corecte, până la momentul curent. Clasificările eronate determină o corecție a vectorului pondere care se face exact ca la algoritmul perceptronului. Clasificările corecte vor determina o schimbare a vectorului din “buzunarul nostru” numai dacă vectorul curent clasifică corect o secvență de forme de intrare de lungime mai mare decât cea realizată de ultimul vector memorat.

Gallant a demonstrat că pe măsură ce numărul de iterații creşte probabilitatea ca vectorul memorat să fie optimal se apropie de unu. Este însă posibil ca numărul de iterații care asigură obținerea unui vector optimal să fie foarte mare. În continuare prezentăm în detaliu algoritmul modificat al perceptronului (Gallant).

P_0 Inițializările

Se alege rata de învățare $c > 0$. Se adoptă arbitrar o primă valoare pentru vectorul pondere $\mathbf{w}$. Componentele acestuia se aleg numere mici nenule.

Se inițializează contorul iterațiilor $k := 0$. Se alege N_0 - numărul maxim de iterații.

Se inițializează vectorul aproximativ de separare din "buzunarul nostru" punându-se $\mathbf{b} := \mathbf{w}$.

Se inițializează numărul formelor de instruire corect clasificate de vectorul $\mathbf{b}$ punând $nb := 0$

P_1 Se repetă paşii **P_2**-**P_6** până când este îndeplinită una dintre condițiile următoare:

i) Vectorul pondere nu s-a schimbat la p paşi consecutivi (deci $n = p$). Reamintim că mulțimea de instruire este formată din perechile $(\mathbf{z}^k, d^k)$ cu $k \in \overline{1,p}$.

ii) $k > N_0$

În cazul i) s-a obținut un vector de separare, iar în al doilea caz rezultă un vector aproximativ de separare.

P_2 Se pune $n := 0$.

P_3 Ajustarea ponderilor

Se prezintă rețelei forma de instruire $\mathbf{z}^k$.

Se inițializează o variabilă locală a, punându-se $a := \begin{cases} 1 \text{ dacă} (\mathbf{w}^k)^T \mathbf{z}^k \le 0 \\ 0 \text{ dacă} (\mathbf{w}^k)^T \mathbf{z}^k > 0 \end{cases}$

Se modifică vectorul pondere folosind regula perceptronului adaptată pentru variabila locală a

$$\mathbf{w}^{k+1} = \begin{cases} \mathbf{w}^k + c\mathbf{z}^k & \text{, dacă } a = 1 \\ \mathbf{w}^k & \text{, dacă } a = 0 \end{cases}.$$

P_4 Dacă $a = 0$ se pune $n := n + 1, k := k + 1$, şi se merge la pasul **P_3**. Dacă $a = 1$ se merge la pasul **P_5**.

P_5 Modificarea conținutului buzunarului

Se compară clasificarea realizată de vectorul $\mathbf{w}^k$ cu cea realizată de vectorul din buzunar $\mathbf{b}$. Se verifică condițiile:

Dacă $n = 0$ se trece la pasul **P_7**.

Dacă $n > nb$, atunci se pune $nb := n$ şi $\mathbf{b} = \mathbf{w}^k$.

P_6 Se pune $k := k + 1$ şi se trece la **P_2**.

Observații: Dacă mulțimea de instruire conține două clase liniar separabile algoritmul Gallant produce un vector de separare exact. Pentru clase neseparabile, chiar pentru probleme de mărime medie, nu este sigur că vom obține un vector optim într-un număr rezonabil de iterații. Convergența spre vectorul optim este asimptotică. Cu toate acestea s-a constatat practic că algoritmul Gallant produce vectori pondere satisfăcători, care aproximează numărul optim de clasificări corecte.

PERCEPTRONUL MULTISTRAT

Deşi aduce unele îmbunătăţiri în instruirea perceptronului pentru mulţimi care conţin clase neseparabile algoritmul Gallant prezintă dezavantajul că furnizează o soluţie (vector pondere de separare) aproximativ.

De asemenea este evident că problemele de instruire care necesită suprafeţe de decizie de o formă complicată, nu mai pot fi rezolvate cu un perceptron cu un singur neuron. Exemplul prezentat pentru funcţia logică XOR furnizează o sugestie pentru depăşirea dificultăţilor perceptronului legate de generarea unor regiuni de decizie cu forme mai complicate, de exemplu de tipul celei prezentate în fig. 2. 7.

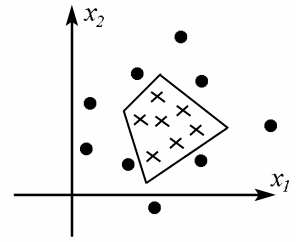

Fig. 2. 7

Soluţia constă în utilizarea unor reţele cu mai multe straturi de neuroni amplasate între câmpul de intrare şi neuronul (stratul de neuroni) de la ieşire. Topologia (arhitectura) unei astfel de reţele numite perceptron multistrat (Multi – Layered Perceptron - MLP) este prezentată în fig. 2. 8. Perceptronul multistrat prezentat în această figură conţine trei straturi cu neuroni activi cu funcţii de activare de tip prag neliniar.

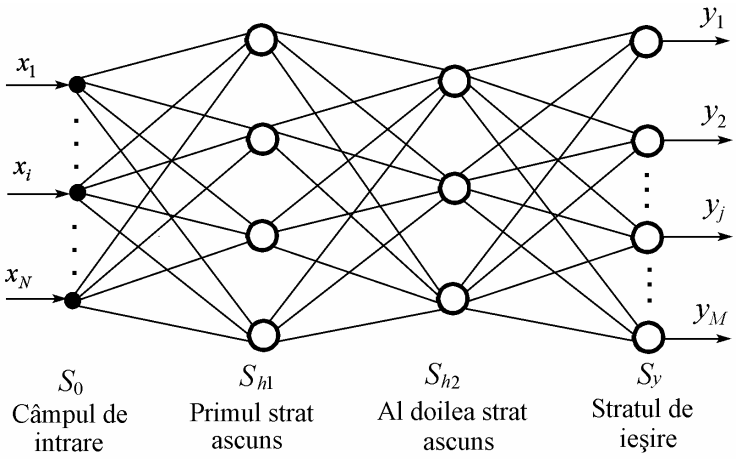

Fig. 2. 8 Arhitectura perceptronului multistrat

Neuronii din primul strat ascuns (S_{h1}) lucrează pe principiul perceptronului standard realizând, fiecare în parte, divizarea (împărţirea) formelor de instruire în două clase separate de un hiperplan. Neuronul va fi activat (are ieşirea 1) doar pentru punctele aflate în partea pozitivă a acestui hiperplan.

Neuronii din al doilea strat ascuns (S_{h2}) generează o regiune de decizie mai complicată, utilizând hiperplanele de separare obţinute de la neuronii primului strat. Într-un mod simplu se poate spune că neuronii stratului S_{h2} realizează intersecţia hiperplanelor generate de neuronii din stratul S_{h1}. Acest lucru se poate obţine dacă aceşti neuroni realizează funcţia logică ŞI. Pentru a vedea cum se poate implementa funcţia logică ŞI să considerăm că stratul S_{h2} conţine *Nh2* neuroni care sunt conectaţi la câmpul S_{h1} prin conexiuni cu ponderile egale cu +1. Adoptându-se valorile de prag (*Nh2-a*), în care $0 < a < 1$, se poate constata uşor că fiecare neuron din S_{h2} are ieşirea +1 doar dacă ieşirile tuturor neuronilor

din primul strat sunt egale cu +1. Neuronii din stratul de ieşire sunt necesari dacă avem mai multe regiuni de decizie convexe disconexe (diferite). Fiecare neuron din acest strat (S_y) realizează funcția SAU logic. Pentru a obține funcția logică SAU se adoptă toate ponderile spre neuronii din stratul de ieşire egale cu 1 şi valorile de prag ale acestora mai mici decât +1. În acest fel, la activarea oricărui neuron din stratul precedent S_{h2}, ieşirile neuronilor din câmpul S_y vor lua valoarea +1. În continuare se vor expune câteva considerații referitoare la problema estimării numărului de neuroni într-un perceptron cu trei straturi (de tipul celui prezentat în fig. 2. 8).

Estimarea numărului de neuroni necesari

Fie N dimensiunea formelor de intrare. Numărul de neuroni din cel de al doilea strat trebuie să fie mai mare ca unu când regiunile de decizie sunt disconexe sau au goluri, neputând fi reduse la o singură regiune convexă. Nu este însă necesar ca numărul neuronilor din stratul S_{h2} să fie mai mare decât numărul de regiuni disconexe din spațiul formelor de intrare (numărul de regiuni care trebuie invățate).

In ceea ce priveşte primul strat, S_{h1}, se apreciază că numărul de neuroni, trebuie să fie suficient de mare pentru ca acest câmp să furnizeze trei sau mai multe laturi pentru fiecare regiune convexă identificată de al doilea strat, S_{h2}. Deci, numărul de neuroni al stratului S_{h1} trebuie să fie de peste trei ori mai mare decît numãrul neuronilor din S_{h2}. Evident, aceasta este o estimare euristică a numărului de neuroni din S_{h1}.

Dacă numărul de neuroni dintr-un strat este mai mic decât cel necesar, atunci rețeaua nu poate construi o reprezentare completă a distribuției vectorilor de intrare şi, deci, nu poate învăța întreaga informație conținută in aceşti vectori.

Dacă numărul neuronilor din oricare strat este prea mare, atunci în rețea se poate genera zgomot. Rețeaua nu va putea construi o reprezentare compactă a relațiilor existente între formele de intrare. Pe de altă parte, robustețea rețelei poate fi obținută doar permițând o anumită redundanță referitoare la numărul neuronilor. În practică, cea mai bună arhitectură a rețelei se poate determina doar prin încercări.

În cazul în care problema de instruire presupune partiționarea spațiului vectorilor de intrare în M clase, perceptronul trebuie să aibă M ieşiri, fiecare ieşire corespunzând unei clase. Presupunem că rețeaua a fost instruită. Când se prezintă rețelei un vector-formă, această forma este asignată (asociată) clasei ce corespunde neuronului de ieşire având cea mai mare valoare a activării.

În unele situații se pot rezolva probleme de instruire complexe prin creşterea numărului de câmpuri ascunse. In realitate aceasta nu este o soluție universală. Pentru unele probleme este necesară o creştere exponențială a numărului de straturi pentru a obține o creştere liniară a vitezei de invățare. De asemenea, există probleme pentru care viteza de instruire descreşte cu mărirea numărului de straturi. Mărirea complexității rețelei este, aşadar, o cale problematică în abordarea proceselor de instruire. O cale diferită este considerarea altor tipuri de neuroni. Acest lucru revine, in esență, la adoptarea unor tipuri diferite de neliniarități in comportarea neuronilor.

Găsirea unor tipuri adecvate de neliniaritate rețele1or neuronale conferă forță de calcul superioară. Comportarea neliniară a neuronilor este esențială pentru un perceptron cu mai multe straturi. Dacă funcția de ieşire ar fi liniară, considerarea mai multor straturi nu ar aduce nici un avantaj — se poate întotdeauna găsi un perceptron cu un singur strat care să realizeze aceeaşi funcție ca şi perceptronul multistrat liniar.

Utilizarea funcțiilor de ieşire neliniare de tip prag prezintă un dezavantaj legat de faptul că aceste funcții nu sunt derivabile. Prin urmare, funcțiile de tip prag nu pot fi utilizate pentru

determinarea ponderilor prin metode de optimizare standard. Acest neajuns poate fi depăşit considerând funcții de ieşire (neliniare) de tip sigmoidal. În acest caz instruirea rețelei se poate face, de exemplu, cu ajutorul *algoritmului de propagare înapoi.*