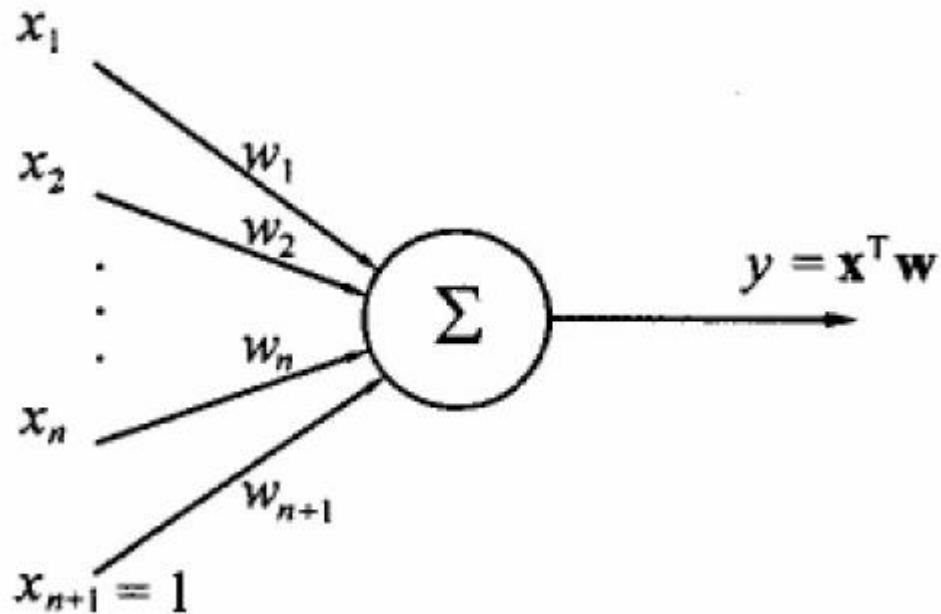


## Laborator 8

Algoritmul de învățare Widrow Hoff bazat pe regula  $\mu-LMS$

Această regulă a fost inițial utilizată pentru a antrena unități liniare de forma:



Algoritmul de învățare bazat pe regula  $\mu-LMS$  constă în determinarea vectorului  $w^*$  folosind metoda gradientului descendent care minimizează eroarea pătratică (SSE – sum squared error).

Fie mulțimea de antrenare  $\{(x^1, d^1), (x^2, d^2), \dots, (x^m, d^m)\}$ .

Funcția criteriu a sumei pătratelor erorilor (SSE) este:

$$J(w) = \frac{1}{2} \sum_i [d^i - (x^i)^T w - b]^2 = \frac{1}{2} \sum_i (d^i - y^i)^2, \text{ unde } y^i = (x^i)^T w + b$$

Regula gradientului descendent de tip batch care minimizează funcția criteriu  $J(w)$  este următoarea:

$$w^{k+1} = w^k + \mu \sum_{i=1}^m (d^i - y^i) x^i$$

$$b^{k+1} = b^k + \mu \sum_{i=1}^m (d^i - y^i)$$

Cum în problemele practice avem  $m > n+1$  este imposibilă satisfacerea tuturor cerințelor  $(x^k)^T w = d^k \quad k = \overline{1, m}$ , deci regula de învățare batch LMS de mai sus nu duce la găsirea unei soluții astfel încât eroarea să fie 0, însă pentru  $\mu$  suficient de mic ( de exemplu,  $\mu = 0.005$ ),  $w^k$  va converge către minimul global  $w^*$  al funcției criteriu  $J(w)$ .

Varianta incrementală a regulei de învățare de mai sus este dată prin:

$$w^{k+1} = w^k + \mu (d^i - y^i) x^i$$

$$b^{k+1} = b^k + \mu (d^i - y^i)$$

S-a demonstrat că pentru  $0 < \mu < \frac{2}{\max_{i=1, m} \|x^i\|}$  acest algoritm converge la soluția,  $w^*$ ,

care minimizează funcția criteriu într-un număr finit de pași.

Astfel, folosind acești algoritmi de învățare este posibil să obținem un vector  $w^*$  care să nu poată clasifica corect o mulțime de antrenare. Acest fapt nu ar trebui să ne surprindă, întrucât se dorește minimizarea funcției criteriu și nu a ratei erorii de clasificare.

Minimul global al funcției criteriu este dat de formula:  $w^* = (XX^T)^{-1} Xd$ , unde  $X = [x^1 \ x^2 \ \dots \ x^m]$  și  $d = [d^1 \ d^2 \ \dots \ d^m]^T$ .

De asemenea, acest algoritm poate fi folosit pentru a determina vectorul ponderilor unei rețele de tip perceptron care să conducă la clasificarea corectă a mulțimii de antrenare.

### Aplicație 1:

Fie mulțimea de antrenare următoare:

$$S = \{([-2,2]^T, -1), ([-2,3]^T, -1), ([-1,1]^T, -1), ([-1,4]^T, -1), ([0,0]^T, -1), ([0,1]^T, -1), ([0,2]^T, -1), ([0,3]^T, -1), ([1,0]^T, +1), ([1,1]^T, -1), ([2,1]^T, +1), ([2,2]^T, -1), ([3,-1]^T, +1), ([3,0]^T, +1), ([3,1]^T, +1), ([3,2]^T, +1), ([4,-2]^T, +1), ([4,1]^T, +1), ([5,-1]^T, +1), ([5,0]^T, +1)\}.$$

1) Implementați în MATLAB algoritmul de învățare Widrow-Hoff folosind regula de învățare  $\mu$ -LMS varianta batch (antrenați rețeaua folosind 100 de pași) și varianta incrementală alegând exemplele din mulțimea de antrenare aleator (antrenând rețeaua în acest caz în 2000 de epoci) și plotați dreapta de separare obținută pe baza lui  $w$  și a lui  $b$  pentru mulțimea de antrenare  $S$  de mai sus, respectiv punctele din mulțimea de antrenare. Observație: Considerăm rata de învățare  $\mu = 0.005$ , vectorul ponderilor inițial  $w^1 = [0 \ 0]^T$  și utilizarea unei rețele fără bias.

2) Studiați convergența celor doi algoritmi de la punctul 1), plotând pe axa Ox pașii de antrenare versus valorile  $\|w^k - w^*\|^2$  corespunzătoare pe axa Oy (adică pentru vectorii de ponderi obținuți la iterația  $k$ ) pentru fiecare algoritm în parte, unde  $w^*$  reprezintă soluția care minimizează funcția criteriu SSE. Interpretați rezultatele. Indicație: se vor folosi axe logaritmice pe axa Ox (vezi semilogx).

3) Punctele a) și b) considerând în procesul de antrenare exemplele din mulțimea de antrenare într-o ordine deterministă pentru algoritmului LMS varianta incrementală.

4) Reprezentați grafic suprafața de eroare asociată funcției criteriu  $J(w)$  în funcție de ponderile  $w_1$  și  $w_2$  și studiați convergența algoritmilor către vectorul de ponderi care

minimizează această funcție criteriu, plotând pe această suprafață valoarea erorii date de vectorii de ponderi obținuți după fiecare epocă/iterație a algoritmilor.

Indicație: Se construiește o rețea de puncte în plan în pătratul  $[-0.3, 0.3] \times [-0.3, 0.3]$ , corespunzătoare vectorilor de ponderi  $w$  și se calculează în fiecare punct  $w$  valoarea funcției criteriu ( funcția de eroare ),  $J(w)$ .

### **Aplicație 2:**

Să se construiască o mulțime de antrenare având ca vectori de intrare puncte bidimensionale și etichete reale, astfel încât problema găsirii ponderilor să fie nedeterminată, adică să admită mai multe soluții (vectori de ponderi care minimizează funcția de eroare SSE). Realizați suprafața de eroare asociată funcției de eroare în raport cu ponderile  $w_1$  și  $w_2$  și reprezentați pe aceasta vectorul de ponderi care minimizează funcția SSE obținut folosind newlind și de asemenea șirul vectorilor de ponderi obținuți în antrenarea perceptronului folosind funcția train. Interpretați rezultatele.

### **Aplicație 3:**

Aplicația 2 pentru o mulțime de date de antrenare neliniare.

De exemplu:

$$P = [ 1.0 \ 2.0 \ 3.0]$$

$$T = [ 4.0 \ 5.0 \ 6.0];$$

Studiați evoluția funcției de performanță (a funcției de eroare) după fiecare epocă și eroarea minimă obținută după antrenare.

### **Aplicație 4:**

Definiți o mulțime de antrenare ca în aplicația 2 și construiți un perceptron cu funcție de transfer liniară pe care îl antrenați folosind regula de învățare a lui Widrow Hoff pentru o rată de învățare de două ori mai mare decât cea furnizată de funcția *maxlinlr*. Realizați suprafața de eroare asociată funcției de eroare în raport cu ponderile  $w_1$  și  $w_2$  și reprezentați pe aceasta vectorul de ponderi care minimizează funcția SSE obținut folosind

metoda celor mai mici pătrate ( $w^*$ ) și vectorii de ponderi rezultați după fiecare aplicare a regulii de învățare. Vizualizați rezultatele și comentați-le.