Laborator 8

Algoritmul de învățare Widrow-Hoff bazat pe regula μ-LMS (*least mean squares*) antrenează ponderile unui perceptron cu funcția de transfer liniară (funcția ***purelin*** în Matlab) pentru a minimiza funcția criteriu *J* dată de suma pătratelor erorilor. Pentru mulțimea de antrenare S = $\{(\mathbf{x}^1, d^1), (\mathbf{x}^2, d^2), \ldots, (\mathbf{x}^m, d^m)\}$, algoritmul încearcă să găsească $\mathbf{w}^*$ și b^* astfel încât să minimizeze:

$$J(\boldsymbol{w}, b) = \frac{1}{2}\sum_{i=1}^{m}(d^i - y^i)^2, unde\ \ y^i = \boldsymbol{w}^T\boldsymbol{x}^i + b.$$

Funcția criteriu *J* definește o suprafață de eroare ce ia forma unui paraboloid convex care are punctul de minim global $\boldsymbol{\beta} = \binom{b^*}{\boldsymbol{w}^*}$ soluție a ecuației:

$$\left(\boldsymbol{\beta}^{\boldsymbol{T}}X - \boldsymbol{d}^{\boldsymbol{T}}\right)\left(\boldsymbol{\beta}^{\boldsymbol{T}}X - \boldsymbol{d}^{\boldsymbol{T}}\right)^{\boldsymbol{T}} = 0$$

Rezolvând ecuația de mai sus (detalii la seminar) se obține soluția în formă explicită:

$$\boldsymbol{\beta} = \binom{b^*}{\boldsymbol{w}^*} = (XX^T)^{-1}X\boldsymbol{d},$$

unde X este matricea cu *n+1* linii și *m* coloane (exemplele $\mathbf{x}^i$ sunt vectori coloană cu *n* componente), care pe coloana *i* are forma $[1; \mathbf{x}^i]$, iar $\mathbf{d} = [d^1\ d^2\ \ldots\ d^m]^T$. *Vectorul* $\boldsymbol{\beta}$ *este soluția problemei de regresie liniară pentru mulțimea S.*

O alternativă la calculul soluției optime $\boldsymbol{\beta} = \binom{b^*}{\boldsymbol{w}^*}$ este minimizarea funcției criteriu *J* (este convexă) folosind algoritmul de coborâre pe gradient. Obținem regula de învățare batch dată de:

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

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

Se demonstrează că pentru valori mici ale ratei de învățare μ (spre exemplu 0.005) algoritmul de învățare converge la soluția dată de $\boldsymbol{\beta}$. Soluția $\boldsymbol{\beta}$ pe care o obținem minimizează *J* rezolvând optim problema de regresie liniară multidimensională pe mulțimea de antrenare S. Totuși $\boldsymbol{\beta}$ nu optimizează rata de misclasare, ceea ce înseamnă că soluția obținută s-ar putea să nu separe bine clasele.

Fie mulțimea de antrenare este 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)}. Realizați următoarele:

1. reprezentați punctele mulțimii de antrenare S în plan, asociind markere/culori diferite pentru punctele din cele două clase;
2. aflați soluția $\boldsymbol{\beta} = \begin{pmatrix} b^* \\ \boldsymbol{w}^* \end{pmatrix}$ a ecuației în formă explicită de mai sus și reprezentați dreapta definită de această soluție;
3. implementați algoritmul de învățare Widrow-Hoff varianta batch construind o rețea cu un perceptron cu funcția de transfer *purelin*, funcția de antrenare *trainb* (antrenare de tip batch), regula de învățare pentru ponderi și bias *learnwh* (Widrow-Hoff), rata de învățare 0.005, numărul de epoci 1000. Setați câmpurile corespunzătoare acestor proprietăți într-un obiect Matlab de tip *network* și rulați antrenarea. Reprezentați dreapta de separare obținută.
4. putem înțelege mai bine semnificația soluției $\boldsymbol{\beta} = \begin{pmatrix} b^* \\ \boldsymbol{w}^* \end{pmatrix}$ reprezentând exemplele din mulțimea de antrenare în spațiul 3D asociat (adică primul exemplu de antrenare ($[-2\ ,2]^T$,-1) îl vedem ca punctul 3D (-2,2,-1)) folosind codul Matlab de mai jos:

```
P = [-2 -2 -1 -1 0 0 0 0 1 1 2 2 3 3 3 3 4 4 5 5;...
      2  3  1  4 0 1 2 3 0 1 1 2 -1 0 1 2 -2 1 -1 0;...
     -1 -1 -1 -1 -1 -1 -1 -1 1 -1  1 -1 1 1 1 1 1 1 1 1];
figure, plot3(P(1,:),P(2,:),P(3,:),'sr');hold on;
[x y] = meshgrid(-5:0.5:5);
z = beta(1) + beta(2)*x + beta(3)*y;
figure(gcf),mesh(x,y,z);
```

Soluția $\boldsymbol{\beta} = \begin{pmatrix} b^* \\ \boldsymbol{w}^* \end{pmatrix}$ definește planul în 3D care are proprietatea că predicțiile realizate sunt cele mai apropiate de etichete (avem în acest caz o problemă de regresie).

Pentru exemplul anterior nu putem vizualiza suprafața de eroare definită de funcția criteriu *J* întrucât aceasta depinde de 3 parametri (2 parametri de la **w**, 1 parametru de la **b**). Pentru a putea vizualiza suprafața de eroare precum și evoluția soluției curente $\mathbf{w}^k$ vom restricționa problema numai la 2 parametri (cei dați de **w**), setând o rețea fără bias cu comanda net.biasConect = 0.

5. determinați soluția **w*** pentru cazul în care nu există bias (X este acum o matrice cu *m* linii și *n* coloane, linia *i* conține exemplul $\mathbf{x}^{i\,T}$);
6. implementați algoritmul de învățare Widrow-Hoff varianta batch pentru cazul unei rețele fără bias, păstrând aceeași parametri ca la 3. Reprezentați dreapta de separare obținută.
7. reprezentați paraboloidul dat de funcția criteriu *J(**w**)* pe domeniul **w** = $[w_1,w_2] \in [-0.7, 0.7] \times [-0.7, 0.7]$ folosind funcțiile Matlab *meshgrid* și *surf*.
8. modificați codul Matlab pentru punctul anterior 6 astfel încât să memorați valorile soluției curente $\mathbf{w}^k$ la epoca k. Plotați apoi folosind funcția Matlab *plot3* punctele de forma $(\mathbf{w}^k, J(\mathbf{w}^k))$ observând cum se modifică la fiecare pas soluța curentă $\mathbf{w}^k$ și valoarea funcției criteiu J.
9. reluați punctele 6, 7 și 8 pentru algoritmul de învățare Widrow-Hoff varianta incrementală cu prezentarea aleatoare (funcția *trainr*) a exemplelor la fiecare epocă.
10. reluați punctele 6, 7 și 8 pentru algoritmul de învățare Widrow-Hoff varianta incrementală cu prezentarea ciclică (funcția *trainc*) a exemplelor la fiecare epocă.
11. studiați convergența învățării în cele 3 cazuri (batch, incrementală cu prezentarea aleatoare a exemplelor, incrementală cu prezentarea deterministă a exemplelor) realizând un grafic cu diferența $\|\boldsymbol{w}^k - \boldsymbol{w}^*\|$ după fiecare epocă *k*.