ANTRENAREA SUPERVIZATĂ A REȚELELOR FEEDFORWARD CU FUNCȚII DE ACTIVARE TREAPTĂ ÎN PROBLEME DE CLASIFICARE

1. Considerații generale, motivație şi obiectiv

Rețelele neuronale feedforward cu funcții de activare treaptă pot fi utilizate pentru a realiza clasificarea unei mulțimi date. Complexitatea arhitecturii unei rețele depinde de proprietățile regiunilor ce reprezintă mulțimile de decizie în problema de clasificare. Prin parcurgerea acestei şedințe de aplicații studentul îşi va însuşi cunoştințele necesare proiectării şi antrenării supervizate a rețelelor neuronale în cazul când clasificarea se raportează la regiuni ce pot fi separate liniar. Pentru desfăşurarea practică a experimentelor se va face apel la funcțiile disponibile în ***Neural Networks Toolbox*** care implementează un algoritm numeric standard destinat antrenării acestui tip de rețele.

2. Cunoştințe prealabile necesare

- Capitolul „Antrenarea supervizată a rețelelor neuronale", secțiunea „Rețele feedforward cu funcții de activare treaptă" din cursul "Aplicații ale rețelelor neuronale în automatică".
- Experiență de programare în mediul MATLAB.

3. Breviar teoretic

Utilizarea în probleme de clasificare a rețelelor neuronale feed-forward cu *funcții de activare treaptă*, necesită algoritmi de antrenare specifici, care exploatează forma particulară a funcțiilor de activare. Tot datorită formei particulare a funcțiilor de activare, aceste tipuri de rețele sunt frecvent referite în literatură drept "*rețele perceptron*", întrucât neuronul cu funcție de activare treaptă a fost introdus, inițial, sub denumirea de "*perceptron*".

3.1. Utilizarea rețelelor cu un singur strat

3.1.1. Proprietatea de separare folosită în clasificare

O rețea neuronală cu un singur strat cu n intrări, $\boldsymbol{x} \in \mathbb{R}^n$, p ieşiri (neuroni) $\boldsymbol{y} \in \mathbb{R}^p$ şi funcții de activare treaptă posedă proprietatea de a partiționa spațiul vectorilor de intrare (adică $\mathbb{R}^n$) prin *p hiperplane* (adică de a împărți acest spațiu în 2^p mulțimi). Astfel rețeaua va putea fi utilizată în probleme de clasificare *numai dacă* regiunile de decizie sunt mulțimi din $\mathbb{R}^n$ separabile liniar (adică cu ajutorul unor hiperplane).

3.1.2. Obiectivul antrenării rețelei

La intrarea rețelei sunt prezentate *N eşantioane*, sau *prototipuri* (eng. *pattern*), notate $\boldsymbol{x}_i \in \mathbb{R}^n$, $i = 1,\ldots,N$, despre care *se presupune că* aparțin claselor C_j, $j = 1,\ldots,2^p$, clase ce se pot defini ca regiuni separabile liniar din $\mathbb{R}^n$.

Să introducem, pentru simplificarea scrierii ulterioare, notația $M = 2^p$. Evident problema clasificării are sens pentru $N > M$, altminteri situându-ne într-o situație pur banală.

În termeni concreți, pentru fixare, vom considera situațiile următoare:

- m_1 vectori din $\mathbb{R}^n$ aparțin clasei C_1, adică:

$$\boldsymbol{x}_k \in C_1,\ k = 1,\ldots,m_1; \tag{1-1}$$

- m_2 vectori din $\mathbb{R}^n$ aparțin clasei C_2, adică:

$$\boldsymbol{x}_k \in C_2,\ k = m_1+1,\ldots,m_1+m_2; \tag{1-2}$$

$\vdots$

- m_M vectori din $\mathbb{R}^n$ aparțin clasei C_M, adică:

$$\boldsymbol{x}_k \in C_2,\ k = m_1+m_2+\ldots+m_{M-1}+1,\ldots,m_1+m_2+\ldots+m_{M-1}+m_M = N; \tag{1-M}$$

Modalitățile de definire prin expresii analitice a celor $M = 2^p$ clase (adică ecuațiile celor *p* hiperplane ce realizează separarea) *nu sunt cunoscute aprioric*. Determinarea sub formă analitică a acestor hiperplane reprezintă tocmai *obiectivul antrenării rețelei*, astfel încât, în urma unei antrenări adecvate, parametrii rețelei (ponderi şi deplasări) să furnizeze coeficienții ecuațiilor ce definesc hiperplanele de separare.

În acest scop, fiecăreia dintre cele 2^p clase C_j i se ataşează câte un vector $\underline{\boldsymbol{y}}_j \in \mathbb{R}^p$, $j = 1,\ldots,2^p$, cu elemente binare, astfel:

- valori 0 şi 1 – în cazul când funcțiile de activare ale neuronilor sunt de tip treaptă unipolară;
- valori –1 şi 1 – în cazul când funcțiile de activare ale neuronilor sunt de tip treaptă bipolară.

Subliniem faptul că prin acest procedeu se construiesc exact 2^p vectori distincți, cu elemente binare, ceea ce permite realizarea unei corespondențe biunivoce între vectorii $\underline{\boldsymbol{y}}_j$ şi clasele C_j. Astfel, problema clasificării revine la a antrena rețeaua (adică a determina parametrii rețelei) de aşa manieră încât *rețeaua cu parametrii rezultați din antrenare* să asigure satisfacerea următoarelor *N* egalități ce decurg din (1-1)...(1-M):

$$\boldsymbol{f}(\boldsymbol{x}_k) = \begin{cases} \underline{\boldsymbol{y}}_1, & k = 1,\ldots,m_1; \\ \underline{\boldsymbol{y}}_2, & k = m_1+1,\ldots,m_1+m_2; \\ \ldots \\ \underline{\boldsymbol{y}}_M, & k = m_1+\ldots+m_{M-1}+1,\ldots,N; \end{cases} \tag{2}$$

Cu alte cuvinte, fiecărui vector prototip $\boldsymbol{x}_i$, $i=1,\ldots,N$, i se poate ataşa un vector țintă (eng. *target*) $\underline{\boldsymbol{z}}_i$, cu:

$$\underline{\boldsymbol{z}}_i \in \left\{\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_M\right\},\ i=1,\ldots,N, \tag{3}$$

astfel încât să fie satisfăcute egalitățile (2). Subliniem faptul că dintre cei N vectori țintă $\underline{\boldsymbol{z}}_i$, $i=1,\ldots,N$, numai $M<N$ sunt distincți (coincizând cu unul din vectorii $\underline{\boldsymbol{y}}_1, \underline{\boldsymbol{y}}_2, \ldots, \underline{\boldsymbol{y}}_M$). Astfel, prin antrenare, se va urmări ca:

$$\boldsymbol{f}(\boldsymbol{x}_i) = \underline{\boldsymbol{z}}_i,\ i=1,\ldots,N, \tag{4}$$

unde forma particulară a lui $\underline{\boldsymbol{z}}_i$ din mulțimea (3) este precizată prin egalitatea (2), corespunzător valorii concrete a indicelui i.

Observații

1° În formularea obiectivului antrenării s-a presupus că se operează cu clase separabile liniar. Totuşi, uzual, în practică, această ipoteză de separabilitate liniară *nu poate fi verificată* înainte de a începe antrenarea. Altfel spus, nu dispunem de un procedeu care să ne asigure, *înainte de efectuarea antrenării*, că vectorii de intrare $\boldsymbol{x}_i$, $i=1,\ldots,N$, pot fi grupați cu satisfacerea condițiilor (1-1),..,(1-*M*), în $M=2^p$ clase C_j, $j=1,\ldots,M$, separabile liniar. Astfel, *însuşi rezultatul antrenării* este cel care *confirmă* sau *infirmă* (prin verificarea satisfacerii celor N egalități (4)) ipoteza separabilității liniare a claselor, iar în caz de confirmare, sunt furnizate şi valorile numerice ale coeficienților ecuațiilor pentru hiperplanele de separare.

2° În cazul când rezultatul antrenării arată că cele N egalități (4) nu pot fi satisfăcute, nu înseamnă că vectorii $\boldsymbol{x}_i$, $i=1,\ldots,N$, nu *ar putea* fi grupați în $M=2^p$ *clase oarecare*, separabile liniar, ci semnifică faptul că *modul concret de asignare* a lui $\boldsymbol{x}_i$ la C_j definit prin relațiile (1-1),...,(1-*M*) *nu permite* separarea liniară a claselor. Deci, *un alt mod de definire* a relațiilor de apartenență (1-1),...,(1-M) *ar putea* (nu în mod sigur) conduce la clase separabile liniar.

3.1.3. Algoritmul de antrenare al rețelei

Algoritmul de *antrenare*, *învățare*, sau *instruire* (eng. *training*, *learning*) constă în actualizarea parametrilor rețelei, adică a ponderilor (elementele matricei $\boldsymbol{W}\in\mathbb{R}^{p\times n}$) şi a deplasărilor (elementele vectorului coloană $\boldsymbol{b}\in\mathbb{R}^p$), cu scopul de a satisface cele N egalități (4).

Pentru algoritmul de antrenare se stabileşte de către utilizator un *număr maxim de iterații* sau *epoci*. La *fiecare iterație* a algoritmului de antrenare se parcurg etapele descrise mai jos. Valorile parametrilor rețelei la începerea unei iterații sunt notate prin $\boldsymbol{W}_{old}\in\mathbb{R}^{p\times n}$ (pentru ponderi) şi $\boldsymbol{b}_{old}\in\mathbb{R}^p$ (pentru deplasări).

Etapa 1 (Etapa de prezentare)

Se prezintă *vectorii de intrare* $\boldsymbol{x}_i$, $i=1,\ldots,N$, sub forma matricei:

$$\boldsymbol{X}=[\boldsymbol{x}_1\ldots\boldsymbol{x}_N]\in\mathbb{R}^{n\times N}. \tag{5}$$

Pentru *valorile curente* ale parametrilor rețelei ($\boldsymbol{W}_{\text{old}}\in\mathbb{R}^{p\times n}$, $\boldsymbol{b}_{\text{old}}\in\mathbb{R}^{p}$) se calculează ieşirile rețelei:

$$z_i=\boldsymbol{f}(\boldsymbol{x}_i)=\sigma\left(\boldsymbol{W}_{\text{old}}\cdot\boldsymbol{x}_i+\boldsymbol{b}_{\text{old}}\right),\ i=1,\ldots,N, \tag{6}$$

care se organizează sub forma matricei:

$$\boldsymbol{Z}=[z_1\ldots z_N]\in\mathbb{R}^{p\times N}. \tag{7}$$

Etapa 2 (Etapa de verificare)

Se calculează vectorii erorilor:

$$\boldsymbol{e}_i=\underline{z}_i-z_i\in\mathbb{R}^{p},\ i=1,\ldots,N, \tag{8}$$

ca diferențe dintre vectorii țintă $\underline{z}_i$ (aparținând mulțimii (3)) şi vectorii ieşirii la iterația curentă z_i (definiți conform (6)).

Algoritmul se opreşte dacă este îndeplinită una din următoarele condiții:
(χ1) toți vectorii eroare sunt nuli, adică:

$$\boldsymbol{e}_i\equiv\boldsymbol{0},\ i=1,\ldots,N; \tag{9}$$

sau:
(χ2) a fost atins numărul maxim de iterații.

La oprire, algoritmul va furniza valorile parametrilor rețelei rezultate din antrenare, care vor fi notate prin $\boldsymbol{W}_f\in\mathbb{R}^{p\times n}$ (ponderi) şi $\boldsymbol{b}_f\in\mathbb{R}^{p}$ (deplasări).

Dacă nici una din condițiile (χ1) sau (χ2) nu sunt îndeplinite, se continuă cu etapa următoare a iterației curente.

Etapa 3 (Etapa de actualizare a parametrilor)

Se construieşte matricea erorilor:

$$\boldsymbol{E}=[\boldsymbol{e}_1\ldots\boldsymbol{e}_N]\in\mathbb{R}^{p\times N}, \tag{10}$$

cu ajutorul căreia se calculează matricele de actualizare a parametrilor:

- pentru ponderi:

$$\boldsymbol{D}_W=\boldsymbol{E}\cdot\boldsymbol{X}^T\in\mathbb{R}^{p\times n}, \tag{11-a}$$

- pentru deplasări:

$$\boldsymbol{D}_b=\boldsymbol{E}\cdot\underbrace{[1\ldots1]}_{N\text{ elemente}}^{T}\in\mathbb{R}^{p}. \tag{11-b}$$

Se calculează noile valori ale parametrilor:

- pentru ponderi:

$$\boldsymbol{W}_{\text{new}}=\boldsymbol{W}_{\text{old}}+\boldsymbol{D}_W, \tag{12-a}$$

- pentru deplasări:

$$\boldsymbol{b}_{\text{new}} = \boldsymbol{b}_{\text{old}} + \boldsymbol{D}_b \,. \tag{12-b}$$

Cu aceste calcule se încheie iterația curentă şi se trece la o nouă iterație efectuând atribuirile $\boldsymbol{W}_{\text{new}} \to \boldsymbol{W}_{\text{old}}$, $\boldsymbol{b}_{\text{new}} \to \boldsymbol{b}_{\text{old}}$.

Observații

1° Inițializarea parametrilor rețelei (adică $\boldsymbol{W}_{\text{old}}$ şi $\boldsymbol{b}_{\text{old}}$ la prima iterație a algoritmului) se face cu valori arbitrare.

2° Condiția (χ1) din Etapa 2 (anularea tuturor vectorilor eroare) poate fi exprimată unitar, prin intermediul *erorii pătratice globale* (eng. *sum squared error*), sau a *erorii medii absolute* (eng. *mean absolute error*), $SSE = \sum_{i=1}^{N} \boldsymbol{e}_i^T \boldsymbol{e}_i$, respectiv $MAE = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{p}\sum_{k=1}^{p} |\, \boldsymbol{e}_{i,k} \,|$, (unde $\boldsymbol{e}_{i,k}$ notează componenta k a vectorului eroare $\boldsymbol{e}_i = [\boldsymbol{e}_{i,1} \ldots \boldsymbol{e}_{i,p}]^T \in \mathbb{R}^p$) sub forma:

$$SSE = 0 \,, \tag{13}$$

respectiv:

$$MAE = 0 \,, \tag{13'}$$

care sunt echivalente cu (9). Datorită simplității, în programare se preferă utilizarea unui test de forma (13) sau (13'), față de cele N teste formulate în (9).

3° Este demonstrat faptul că vectorii $\boldsymbol{x}_i$, $i = 1, \ldots, N$, pot fi grupați în clasele C_j, separabile liniar, conform (1-1), (1-2), ... , (1-M), *dacă şi numai dacă* algoritmul de antrenare va conduce la satisfacerea condiției (χ1) într-un număr finit de iterații. Din acest motiv, la stabilirea numărului maxim de iterații, utilizatorul trebuie să aibă în vedere o valoare acoperitoare pentru satisfacerea condiției (χ1), evident, sub rezerva că (χ1) poate fi îndeplinită.

4° În cazul când condiția (χ1) este satisfăcută, valorile $\boldsymbol{W}_f \in \mathbb{R}^{p \times n}$ şi $\boldsymbol{b}_f \in \mathbb{R}^p$ furnizate în finalul algoritmului de antrenare definesc ecuațiile celor p hiperplane (care separă cele $M = 2^p$ clase) prin intermediul celor p componente scalare ale egalității:

$$\boldsymbol{W}_f \cdot \boldsymbol{x} + \boldsymbol{b}_f = \boldsymbol{0} \,. \tag{14}$$

5° Dacă (χ1) nu poate fi îndeplinită într-un număr suficient de mare de iterații ale algoritmului de antrenare, rezultă că vectorii $\boldsymbol{x}_i$, $i = 1, \ldots, N$, nu pot fi grupați în clasele C_j separabile liniar, conform (1-1), (1-2), ... , (1-M). Este evident că în acest caz eroarea pătratică globală *nu se va anula*.

3.2. Utilizarea rețelelor cu mai multe straturi

Este demonstrat faptul că rețelele cu mai multe straturi permit realizarea clasificării în condițiile când clasele *nu constituie* regiuni separabile liniar din $\mathbb{R}^n$. În aceste situații, complexitatea rețelei (ca şi număr de straturi) este dependentă de proprietățile mulțimilor din $\mathbb{R}^n$ utilizate pentru clasificare (de exemplu, satisfacerea condiției de convexitate). Procedeele de antrenare constituie generalizări ale algoritmului discutat în cazul rețelelor cu un singur strat.

4. Facilităţi software oferite de *Neural Network Toolbox*

Neural Network Toolbox este unul dintre primele pachete de programe MATLAB lansate de The MathWorks, ceea ce demonstrează importanţa arătată domeniului reţelelor neurale de către firma dezvoltatoare a acestui software. Versiunea 4.0 a acestui toolbox a fost lansată în anul 2000, împreună cu versiunea MATLAB Release 12. Odată cu dezvoltarea software-ului de bază (MATLAB), pachetului ***Neural Network Toolbox*** i-au fost aduse modificări minore.

Neural Network Toolbox admite lucrul cu diferite tipuri de reţele neurale. Această flexibilitate se datorează reprezentării orientate obiect a reţelelor, care permite definirea unor arhitecturi diverse şi implementarea algoritmilor specifici acestora. Un obiect de tip network constă din numeroase proprietăţi ce pot fi setate în mod adecvat pentru specificarea arhitecturii şi comportării reţelei neurale şi poate fi creat cu funcţia network. Pentru crearea unor reţele neurale cu o structură standard există însă funcţii special concepute (de exemplu newp, newlin, newlind, newff, newrbf).

Proprietăţile unui obiect network sunt grupate în funcţie de tipul lor în mai multe grupe, şi anume:

- architecture: proprietăţi ce permit setarea numărului de intrări şi de straturi ale reţelei neurale, precum şi modul de conectare a acestora;
- functions: pentru definirea funcţiilor pentru operaţiile de bază cu reţeaua neurală (iniţializare, modul de adaptare a parametrilor, criteriul de performanţă, funcţia de antrenare, precum şi parametri specifici acestora);
- weight and bias values: pentru inspectarea sau setarea valorilor ponderilor şi deplasărilor neuronilor din reţea;
- other: pentru păstrarea altor date dorite de utilizator (de exemplu, comentarii).

În pachetul ***Neural Network Toolbox*** există şi o interfaţă grafică cu utilizatorul (*Graphical User Interface - GUI*), nntool, pentru importarea, crearea, utilizarea şi exportarea reţelelor neurale şi/sau datelor. De asemenea, ***Neural Network Toolbox*** pune la dispoziţia utilizatorilor un set de blocuri pentru construirea reţelelor neurale în Simulink, precum şi funcţia gensim pentru generarea versiunii Simulink a unei reţele neurale oarecare care a fost creată în MATLAB. Modul de utilizare a funcţiilor este ilustrat prin numeroase programe demonstrative ce pot fi lansate în execuţie prin intermediul comenzii demos.

Funcţia newp crează o reţea perceptron cu un singur strat. După ce a fost creată, o reţea poate fi antrenată utilizând funcţia train care implementează algoritmul de antrenare a unei reţele perceptron conform parametrilor de antrenare aleşi după creare. Pentru folosirea algoritmului de antrenare descris în lucrarea de laborator se alege rutina 'trainb' drept funcţie de antrenare a reţelei (net.trainFcn = 'trainb'). Valoarea implicită este net.trainFcn = 'trainc'.

Pentru simularea comportării unei reţelei neurale se poate utiliza funcţia sim.

Pentru reprezentarea grafică a vectorilor de intrare bi- şi tridimensionali se poate apela funcţia plotpv. În scopul ataşării pe acest grafic a hiperplanelor de separare dintre clase se poate apela funcţia plotpc (după un apel prealabil al funcţiei plotpv).

Modul de apel al funcţiilor descrise mai sus poate fi studiat în urma consultării help-ului aferent.

5. Problematica propusă pentru studiu

Problema 1

Se consideră un perceptron cu două intrări care se utilizează pentru a clasifica următorii 5 vectori bidimensionali:

$$\boldsymbol{x}_1 = \begin{bmatrix} 0.3 \\ -0.4 \end{bmatrix}, \boldsymbol{x}_2 = \begin{bmatrix} -0.4 \\ -0.5 \end{bmatrix}, \boldsymbol{x}_3 = \begin{bmatrix} 0.5 \\ 1.5 \end{bmatrix}, \boldsymbol{x}_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \boldsymbol{x}_5 = \begin{bmatrix} -0.3 \\ 0.5 \end{bmatrix},$$

în două clase C_1,C_2 astfel:

$$\boldsymbol{x}_1 \in C_2,\ \boldsymbol{x}_2 \in C_1,\ \boldsymbol{x}_3 \in C_2,\ \boldsymbol{x}_4 \in C_1,\ \boldsymbol{x}_5 \in C_1.$$

Să se elaboreze un program MATLAB care să creeze şi să antreneze perceptronul pentru un număr de iterații stabilit de către utilizator. Se vor reprezenta grafic vectorii de intrare şi se va trasa dreapta definită prin parametrii perceptronului antrenat. În finalul antrenării se va verifica, prin simulare, dacă rețeaua antrenată permite realizarea clasificării.

Se vor raporta următoarele rezultate ale experimentelor:

(*i*) valorile parametrilor inițiali ai rețelei;

(*ii*) valorile parametrilor rețelei la sfârşitul antrenării;

(*iii*) reprezentarea grafică a vectorilor de intrare şi dreapta de separare a claselor definită de rețea, în urma antrenării;

(*iv*) numărul de iterații cât a durat antrenarea;

(*v*) reprezentarea grafică a erorii pătratice globale ca funcție de numărul de iterații parcurse.

Problema 2

Se va relua Problema 1, pentru cazul când se urmăreşte clasificarea celor 5 vectori astfel:

$$\boldsymbol{x}_1 \in C_2,\ \boldsymbol{x}_2 \in C_1,\ \boldsymbol{x}_3 \in C_2,\ \boldsymbol{x}_4 \in C_1,\ \boldsymbol{x}_5 \in C_2.$$

Problema 3

Se consideră o rețea perceptron cu un singur strat cu doi neuroni, având două intrări, care se utilizează pentru a clasifica următorii 7 vectori bidimensionali:

$$\boldsymbol{x}_1 = \begin{bmatrix} 0.8 \\ 1.6 \end{bmatrix}, \boldsymbol{x}_2 = \begin{bmatrix} 1.25 \\ 1 \end{bmatrix}, \boldsymbol{x}_3 = \begin{bmatrix} 0.3 \\ 0.5 \end{bmatrix}, \boldsymbol{x}_4 = \begin{bmatrix} -1 \\ -1.25 \end{bmatrix}, \boldsymbol{x}_5 = \begin{bmatrix} 0. \\ 0.2 \end{bmatrix}, \boldsymbol{x}_6 = \begin{bmatrix} -0.3 \\ 0.8 \end{bmatrix}, \boldsymbol{x}_7 = \begin{bmatrix} -0.5 \\ -1.5 \end{bmatrix},$$

în patru clase C_1, C_2, C_3, C_4 astfel:

$$\boldsymbol{x}_1 \in C_1,\ \boldsymbol{x}_2 \in C_2,\ \boldsymbol{x}_3 \in C_3,\ \boldsymbol{x}_4 \in C_4,\ \boldsymbol{x}_5 \in C_3,\ \boldsymbol{x}_6 \in C_3,\ \boldsymbol{x}_7 \in C_4.$$

Să se elaboreze un program MATLAB care să creeze şi să antreneze rețeaua. Se vor reprezenta grafic vectorii de intrare şi se vor trasa cele două drepte definite prin parametrii rețelei antrenate. În finalul antrenării se va verifica, prin simulare, dacă rețeaua antrenată permite realizarea clasificării.

Se vor raporta următoarele rezultate ale experimentelor:

(*i*) valorile parametrilor inițiali ai rețelei;

(*ii*) valorile parametrilor rețelei la sfârşitul antrenării;

(*iii*) reprezentarea grafică a vectorilor de intrare şi dreptele de separare a claselor definite de rețea, în urma antrenării;
(*iv*) numărul de iterații cât a durat antrenarea;
(*v*) reprezentarea grafică a erorii pătratice globale ca funcție de numărul de iterații parcurse.

Indicație: Se vor considera vectorii țintă:

(a) $z_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, z_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, z_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, z_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix};$

(b) $z_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, z_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, z_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, z_4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$

Ce observați?