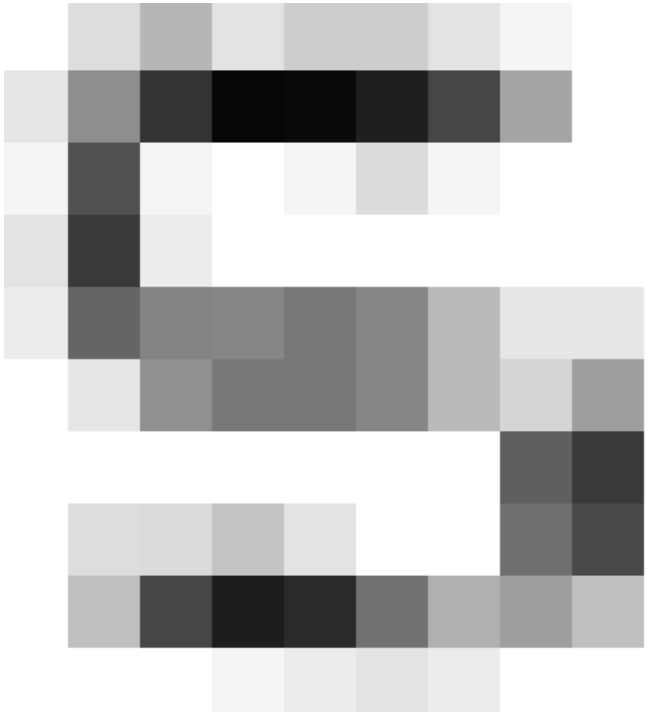
Proiect Tehnici de Optimizare

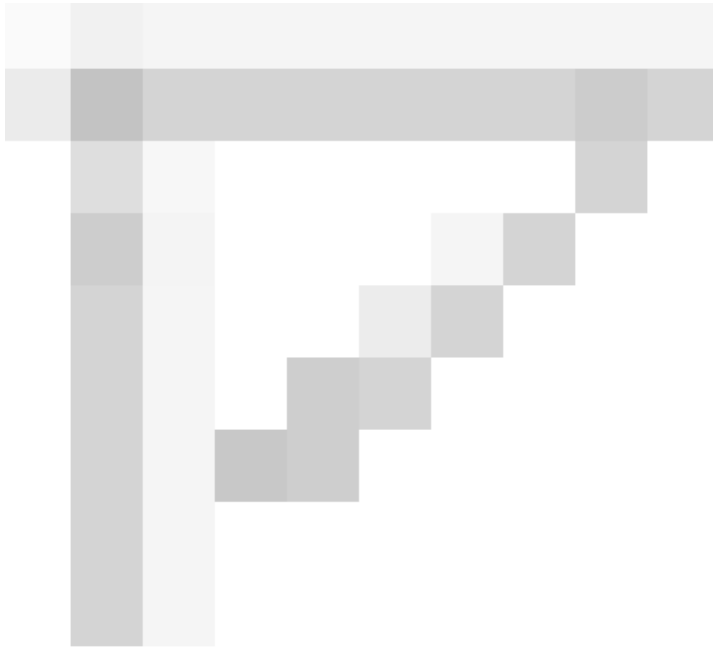
Bondoc Ion-Tudor

Grupa 321AA

RECUNOASTERE DE IMAGINI



Contine “S”



Nu contine “S”

Descrierea Aplicatiei - 1

-> Fiecare poza este construita in formatul 10px * 10px => 100px

-> Un set de imagini este construit astfel incat sa contina litera S

-> Alt set este construit astfel incat sa nu contina litera S

-> Fiecare poza este citita in MATLAB si stocata in cadrul unei matrici, iar matricea este convertita intr-un vector linie

-> Informatia pentru fiecare pixel este un numar de la 0 la 255 (UINT8), 255 insemnand alb iar 0 negru complet

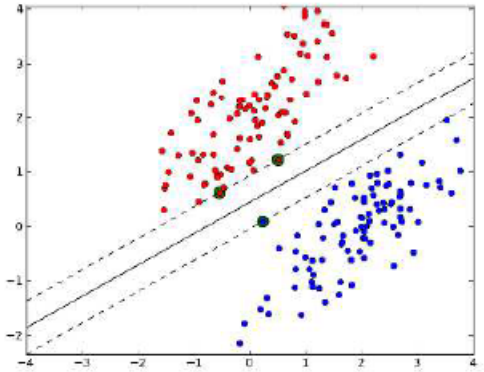
Descrierea Aplicatiei - 2

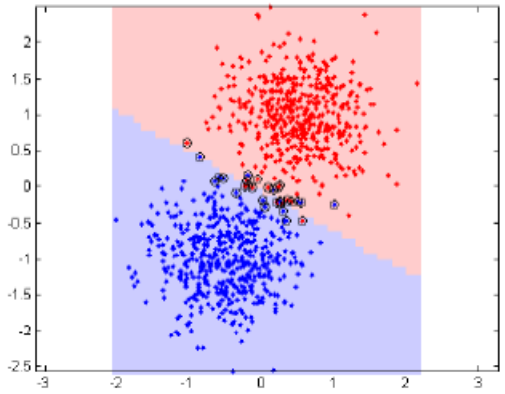
-> Realizam un program care “invata” din cele 2 seturi de imagini si care ulterior decide daca o imagine furnizata ca si test contine sau nu litera “S”

-> Am implementat solutia SVM folosind CVX, dupa care am incercat sa implementam o solutie sub forma Regresiei Logistice (asemanator cu Laboratorul 2)

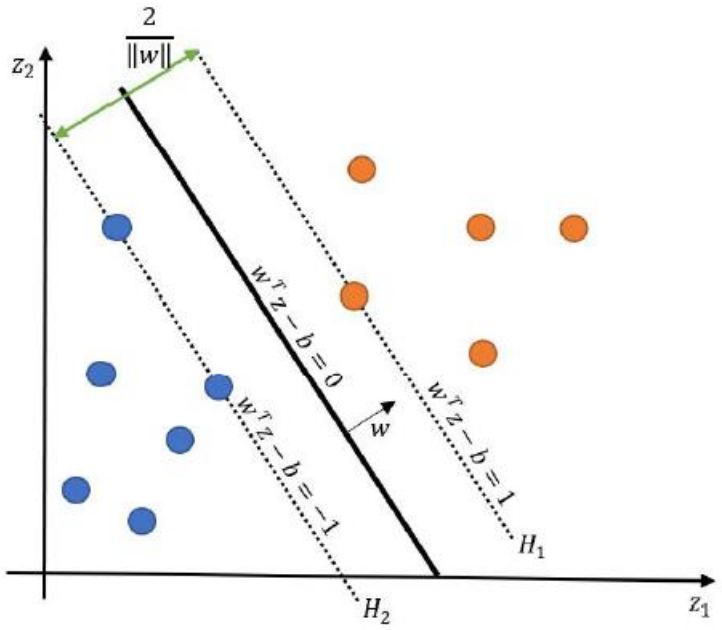
Formularea matematica a problemei - 1

- -> Tehnica SVM se bazeaza pe gasirea unui hiperplan de separare a doua clase de obiecte
- -> Hiperplanul este definit de H = {z ∈ R^n : w' z = b} caracterizat de parametrii w ∈ R^n si b ∈ R.
- -> Trebuie, deci, sa gasim parametrii optimi w si b





Formularea matematica a problemei-2



-> distanta intre cele doua hiperplane este 2 / ||w|| si vrem sa fie maxima

-> reformulam problema astfel incat sa fie QP

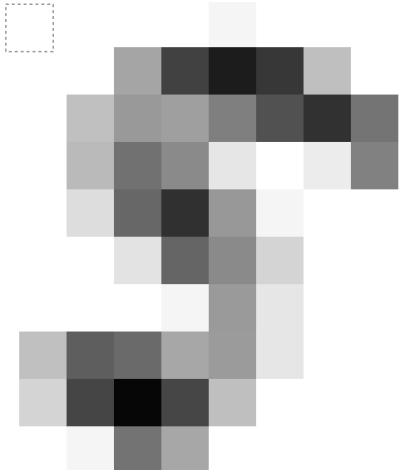
-> variabila de decizie este x=[w' b]'

$$\begin{cases} w^T z_i - b \geq 1, \forall y_i = 1 \\ w^T z_j - b \leq -1, \forall y_j = -1 \end{cases} \Rightarrow \begin{cases} y_i(w^T z_i - b) \geq 1, i = 1 : n_1 \\ y_j(w^T z_j - b) \geq 1, j = 1 : n_2 \end{cases}$$

$$\max_{w,b} \frac{2}{\|w\|} \quad \Leftrightarrow \quad \min_{w,b} \frac{\|w\|^2}{2}$$
$$\text{s.l.: } y(w^T z - b) \geq 1 \qquad \text{s.l.: } y(w^T z - b) \geq 1$$

Solutia implementata

```
%% Imagini care contin diferite variatii ale literei S

A = imread('s1.png'); %citeste imaginea
B1 = rgb2gray(A); %conversie in gray
B1 = reshape(B1.',1,[]); %conversie din matrice in vector linie
A = imread('s2.png');
B2 = rgb2gray(A);
B2 = reshape(B2.',1,[]);
A = imread('s3.png');
B3 = rgb2gray(A);
B3 = reshape(B3.',1,[]);
A = imread('s4.png');
B4 = rgb2gray(A);
B4 = reshape(B4.',1,[]);
A = imread('s5.png');
B5 = rgb2gray(A);
B5 = reshape(B5.',1,[]);
A = imread('s6.png');
B6 = rgb2gray(A);
B6 = reshape(B6.',1,[]);
A = imread('s7.png');
B7 = rgb2gray(A);
B7 = reshape(B7.',1,[]);
A = imread('s8.png');
B8 = rgb2gray(A);
B8 = reshape(B8.',1,[]);
A = imread('s9.png');
B9 = rgb2gray(A);
B9 = reshape(B9.',1,[]);
```



s1.png

```
A = imread('s10.png');
B10 = rgb2gray(A);
B10 = reshape(B10.',1,[]);
A = imread('s11.png');
B11 = rgb2gray(A);
B11 = reshape(B11.',1,[]);
A = imread('s12.png');
B12 = rgb2gray(A);
B12 = reshape(B12.',1,[]);
A = imread('s13.png');
B13 = rgb2gray(A);
B13 = reshape(B13.',1,[]);
C1=[B1;B2;B3;B4;B5;B6;B7;B8;B9;B10;B11;B12;B13];
C1=double(C1);
```

```
%% Imagini random diferite de S

A = imread('n1.png');
B1 = rgb2gray(A);
B1 = reshape(B1.',1,[]);
A = imread('n2.png');
B2 = rgb2gray(A);
B2 = reshape(B2.',1,[]);
A = imread('n3.png');
B3 = rgb2gray(A);
B3 = reshape(B3.',1,[]);
A = imread('n4.png');
B4 = rgb2gray(A);
B4 = reshape(B4.',1,[]);
```


n1.png

```
A = imread('n5.png');
B5 = rgb2gray(A);
B5 = reshape(B5.',1,[]);
A = imread('n6.png');
B6 = rgb2gray(A);
B6 = reshape(B6.',1,[]);
A = imread('n7.png');
B7 = rgb2gray(A);
B7 = reshape(B7.',1,[]);
A = imread('n8.png');
B8 = rgb2gray(A);
B8 = reshape(B8.',1,[]);
A = imread('n9.png');
B9 = rgb2gray(A);
B9 = reshape(B9.',1,[]);
A = imread('n10.png');
B10 = rgb2gray(A);
B10 = reshape(B10.',1,[]);
A = imread('n11.png');
B11 = rgb2gray(A);
B11 = reshape(B11.',1,[]);
A = imread('n12.png');
B12 = rgb2gray(A);
B12 = reshape(B12.',1,[]);
A = imread('n13.png');
B13 = rgb2gray(A);
B13 = reshape(B13.',1,[]);
```

```
C2=[B1;B2;B3;B4;B5;B6;B7;B8;B9;B10;B11;B12;B13];
C2=double(C2);
```

```
%% CVX

cvx_begin
variable w(100);
variable b(1);
minimize ((1/2)*w'*w)
subject to
C1*w-b*ones(13,1)>=ones(13,1); %1*C1*.. >=
C2*w-b*ones(13,1)<=-ones(13,1); % -1*C2*... >=
cvx_end
```

```
%% Testare
disp 'Test CVX'

A=imread('t1.png');
A=rgb2gray(A);
A = reshape(A.',1,[]);
A=double(A);
rez1 = w'*A'-b;
if (rez1 > 0)
    disp 'Prima imagine test contine S'
else
    disp 'Prima imagine test nu contine S'
end

A=imread('t2.png');
A=rgb2gray(A);
A = reshape(A.',1,[]);
A=double(A);
rez2 = w'*A'-b;
if (rez2 > 0)
    disp 'A doua imagine test contine S'
else
    disp 'A doua imagine test nu contine S'
end
```

t1.png

t2.png

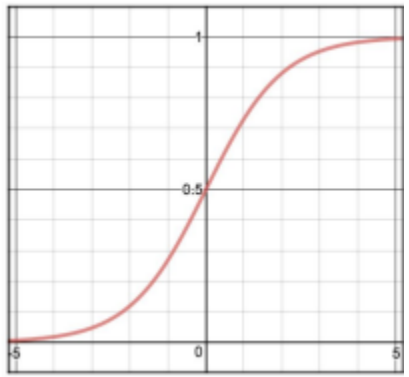
```
Test CVX
Prima imagine test contine S
A doua imagine test nu contine S
```

Regresia logistică

Scop: Modelarea probabilitatii ca o variabila aleatoare(imaginea pe care noi o furnizam) sa apartina hiperplanului in care avem imaginile care contin S sau sa apartina hiperplanului in care avem imaginile care nu contin S. Deci obtinem probabilitatea, si cu ajutorul acesteia mapam imaginea nostra in una dintre cele doua clase.

Functia sigmoid:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Practic, cu ajutorul acestei functii vom realiza maparea intre o predictie(o valoare reala) si o probabilitate(0 sau 1).

Cu ajutorul functiei **sigmoid**, putem afla:

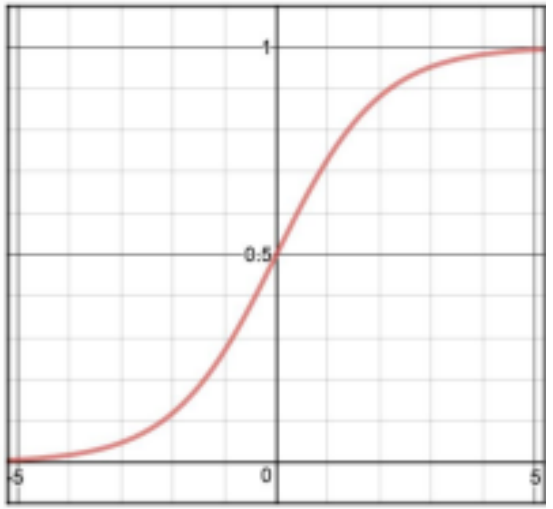
-> probabilitatea ca o imagine sa contina S:

P(class = 1 | X,w) = $\frac{1}{1+e^{-w^T X}} = \sigma(w^T X) = h_w(X)$

->probabilitatea ca o imagine sa nu contina S:

P(class=0|X,w) = 1 - $\frac{1}{1+e^{-w^T X}} = 1 - \sigma(w^T X) = 1 - h_w(X)$

Suma celor doua probabilitati este mereu 1!



Valoarea de prag

$p \geq 0.5, \quad y = 1$
$p < 0.5, \quad y = 0.$

y=class

Practic, daca o valoare a unei probabilitati, P(class|X,w), este >= 0.5 atunci putem face maparea in clasa imaginilor care contin S. Același raționament și pentru cazul invers.

Pe masura ce probabilitatea se apropie de 1, modelul nostru este mai sigur ca observatia se afla in clasa 1.

In cazul nostru:
X_i = [$C_{1,i}$ $C_{2,i}$ $C_{3,i}$...$C_{100,i}$ 1]' %reprezinta datele asociate unei imagini ->**le stim** (sunt date)
w = [w_1 w_2 w_3...w_{101}] %parametrii de regresie-> **trebuie aflate**

Introducem functia **likelihood**:

$$L(w|y,x) = \mathrm{P}(Y|X,w) = \prod_{i=1}^{N} \mathrm{P}\left(y_i|x_i,w\right)$$
$$= \prod_{i=1}^{N} h_w\left(x_i\right)^{y_i}\left(1-h_w\left(x_i\right)\right)^{(1-y_i)}.$$

Aplicăm **log**

Pentru a afla parametrii de regresie(w) trebuie sa **maximizam** urmatoarea functie:

$$\frac{1}{N}\log L(w|y,x) = \frac{1}{N}\sum_{i=1}^{N}\log \mathrm{P}\left(y_i|x_i,w\right)$$
$$= \frac{1}{N}\sum_{i=1}^{N}\left[y^{(i)}\log\left(h_w\left(x^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_w\left(x^{(i)}\right)\right)\right] \qquad (2)$$

Sau **sa minimizam**:

$$F(w) = -\frac{1}{N}\sum_{i=1}^{N}\left[y^{(i)}\log\left(h_w\left(x^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_w\left(x^{(i)}\right)\right)\right]$$
$$= \frac{1}{N}\cdot\left(-y^T\log(h) - (1-y)^T\log(1-h)\right),$$

Acel minus apare pentru ca vom minimiza functia. Stim ca:

$$\max_x f(x) = -\min_x -f(x).$$

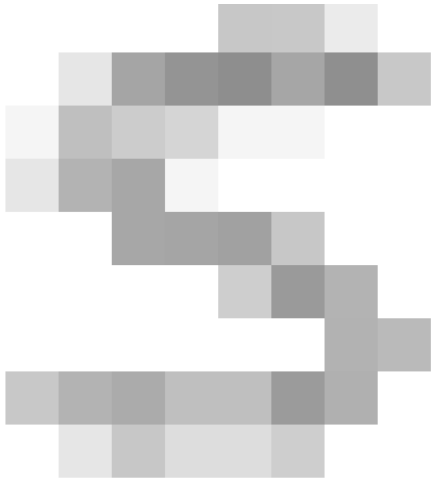
Aplicam Metoda Gradient cu pas ideal

$$\frac{\partial F(w)}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}\left(h_w\left(x_i\right) - y_i\right)x_i^j \quad \Rightarrow \quad \nabla F(w) = \frac{1}{N}(h-y)^T X.$$

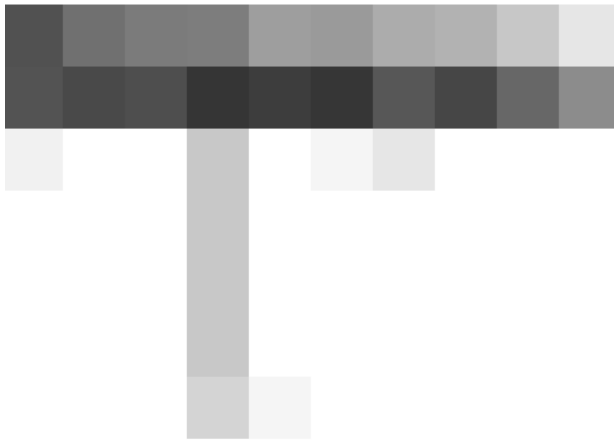
```
%% Regresie logistica

n=26; %nr de poze
C1=C1'; %aduc informatia despre o poza de la forma linie la forma coloana
C2=C2';
pixeli=[C2 C1;ones(1,26)]; %matricea cu pixeli
y=[zeros(13,1);ones(13,1)]; %etichetele
[m,~]=size(pixeli); %nr parametrilor de regresie
epsilon = 0.001; % marja eroarea
maxIter = 100000; % nr maxim de iteratii
wr=zeros(m,1); %w gasit cu regresie logistica

%Metoda Gradient

%Calculam gradientul functiei
F=@(wr)(1/n)*(-y'*log(sigmoid(pixeli'*wr))-(ones(n,1)-y)'*log(1-sigmoid(pixeli'*wr)));
gradient=(1/n)*pixeli*(sigmoid(pixeli'*wr)-y);
```

```
%Metoda Gradient cu pas ideal
k=0;
norma=norm(gradient);
norm_init=norma;
ngpi=[]; %vectorul cu norme pt metoda gradient cu pas constant
ngpi=[ngpi;norma];
while (norma>=epsilon && k<maxIter)
    cost=@(alfa) F(wr-alfa*gradient);
    alfa=fminbnd(cost,0,1); %pasul ideal
    wr=wr-alfa*gradient;
    gradient=(1/n)*pixeli*(sigmoid(pixeli'*wr)-y);
    norma=norm(gradient);
    ngpi=[ngpi;norma];
    k=k+1;
end
wgpi=wr; %salvam w pt metoda gradient pas ideal
```

```
%% Testare Regresie logistica

disp 'Test Regresie Logistica'
prag_decizie=0.5;
A=imread('t3.png');
A=rgb2gray(A);
A = reshape(A.',1,[]);
A=double(A);
A=[A 1];
if (sigmoid(wr'*A') >= prag_decizie)
    disp 'Imaginea contine S';
else
    disp 'Imaginea nu contine S';
end
A=imread('t4.png');
A=rgb2gray(A);
A = reshape(A.',1,[]);
A=double(A);
A=[A 1];
if (sigmoid(wr'*A') >= prag_decizie)
    disp 'Imaginea contine S';
else
    disp 'Imaginea nu contine S';
end
```

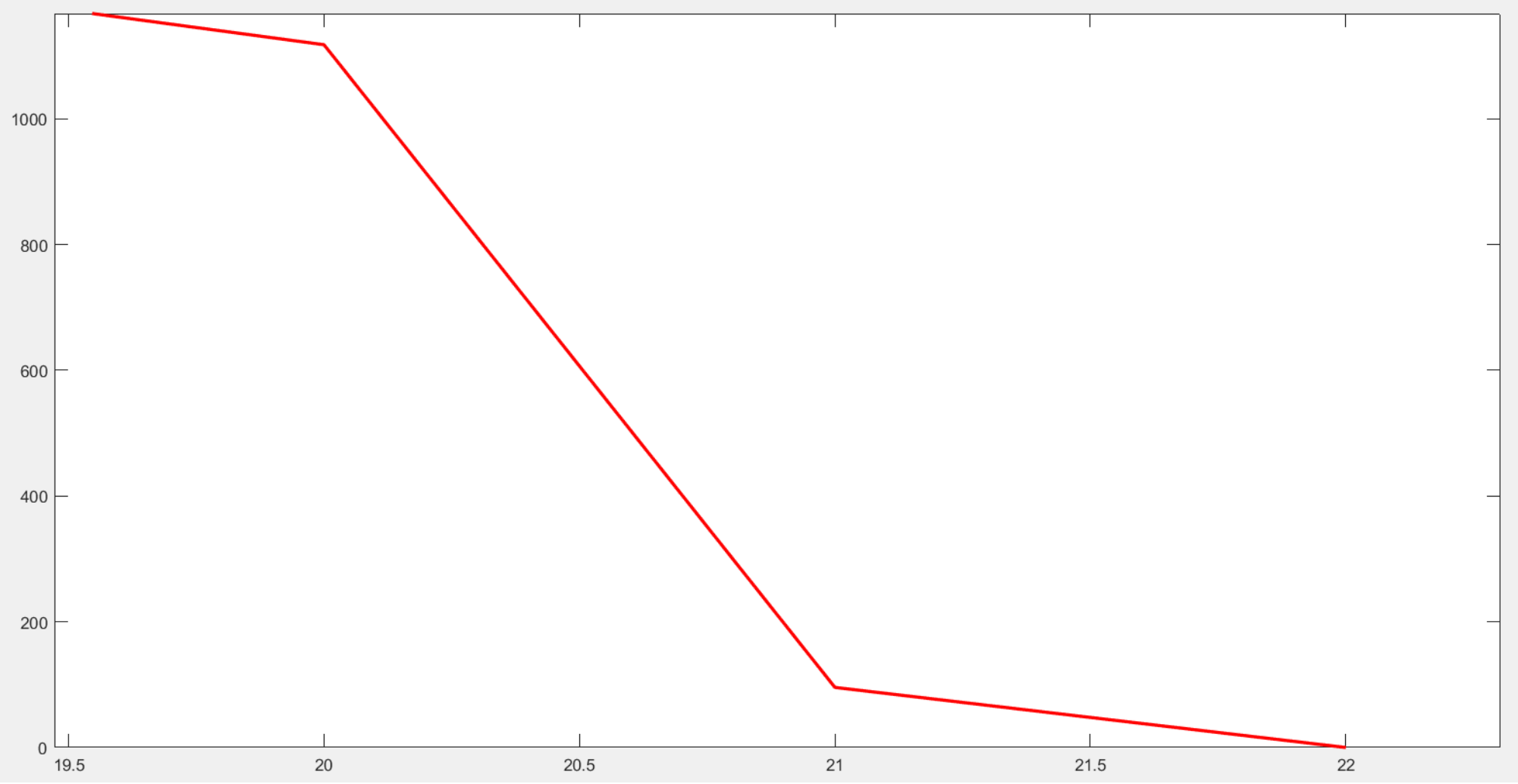


t3.png



t4.png

```
Test Regresie Logistica
Imaginea contine S
Imaginea nu contine S
```



Evolutia normei gradientului

Referinte

- Culegere TO
- Curs
- Indrumar laborator

VA MULTUMIM !