

LAPI – Laboratorul de
Analiza și Prelucrarea
Imaginilor



Universitatea
POLITEHNICA din
București



Facultatea de Electronică,
Telecomunicații și
Tehnologia Informației

TACAI - Tehnici de Analiză și Clasificare Automată a Informației

Note de laborator

Dr.ing. Ionuț Mironică

Conf.dr.ing. Bogdan Ionescu

Laborator 2

Cuprins:

- Introducere în Matlab
- Clasificare de baze de date de imagini
- Prezentare baze de date utilizate
- Exerciții

I. Introducere în Matlab

Cuprins

- Laboratorul cuprinde o scurtă introducere și nu un tutorial complet Matlab:
 - detalii de bază;
 - avantajele și dezavantajele utilizării Matlab în funcție de alte limbaje.

I. Introducere în Matlab

Introducere Matlab

- MATLAB (MATrix LABoratory);
- Reprezintă un limbaj cu mare productivitate pentru dezvoltarea de algoritmi ingineresti;
- Cuprinde algoritmi deja implementați, opțiuni de vizualizare a datelor și unelte pentru o ușoară dezvoltare și proiectare de algoritmi.

I. Introducere în Matlab

Dezavantaje Matlab

- Lent pentru anumite tipuri de probleme și procese;
- Poate fi cu greu implementat în aplicații web;
- Nu este gândit pentru dezvoltarea de sisteme de dimensiuni foarte mari care să ruleze sisteme din producție.

I. Introducere în Matlab

Structură mediu de programare Matlab

The screenshot displays the MATLAB interface with three main windows: Workspace, Command Window, and Command History. The Workspace window shows a list of variables and their properties. The Command Window shows the execution of commands and their output. The Command History window shows a list of previously executed commands.

Workspace / Current Directory

Name	Size	Bytes	Class
C1_4096	1x4	2359536	cell array
C2_4096	1x4	6029552	cell array
Dist1_4096	4096x8x2	3580696	cell array
Dist2_4096	4096x22x2	10731648	cell array
Dist4_4096	4096x70x2	35755056	cell array
Ind4096	4096x7	229376	double array
M4096	1x4096	11264400	cell array
Mout4096	2x4096	22528800	cell array
Neigh4096	4096x8	262144	double array
NeighN2_4096	2x4096	1080464	cell array
NeighN4_4096	4x4096	2951648	cell array
S1_4096	1x4	2359536	cell array
S2_4096	1x4	6029552	cell array
W4096	4096x2	65536	double array
ans	1x2	16	double array
tri4096	8106x3	194544	double array

Command Window

```
>> size(Ind4096,2)
ans =
    7
>> size(Ind4096)
ans =
    4096         7
>> whos
Name           Size           Bytes  Class
C1_4096         1x4             2359536  cell array
C2_4096         1x4             6029552  cell array
Dist1_4096     4096x8x2        3580696  cell array
Dist2_4096     4096x22x2      10731648  cell array
Dist4_4096     4096x70x2      35755056  cell array
Ind4096        4096x7          229376   double array
M4096          1x4096         11264400  cell array
Mout4096       2x4096         22528800  cell array
Neigh4096      4096x8          262144   double array
NeighN2_4096   2x4096         1080464  cell array
NeighN4_4096   4x4096         2951648  cell array
S1_4096        1x4             2359536  cell array
S2_4096        1x4             6029552  cell array
W4096          4096x2          65536    double array
ans            1x2              16        double array
tri4096        8106x3         194544   double array
Grand total is 8186522 elements using 105422504 bytes
>>
```

Command History

```
W4096( (NeighN2_4096(1,1),1), (NeighN2_4096(1,1),1) )
plot(W4096(NeighN2_4096(1,1),1), W4096(NeighN2_4096(1,1),1))
hold on
plot(W4096(1,1), W4096(1,1), 'r.')
plot(W4096(1,1), W4096(1,2), 'r.')
size(Ind4096)
size(Ind4096,2)
size(Ind4096,2)
size(Ind4096)
```

I. Introducere în Matlab

Structură mediu de programare Matlab

Command Window

- Generare de comenzi simple;

Current Directory

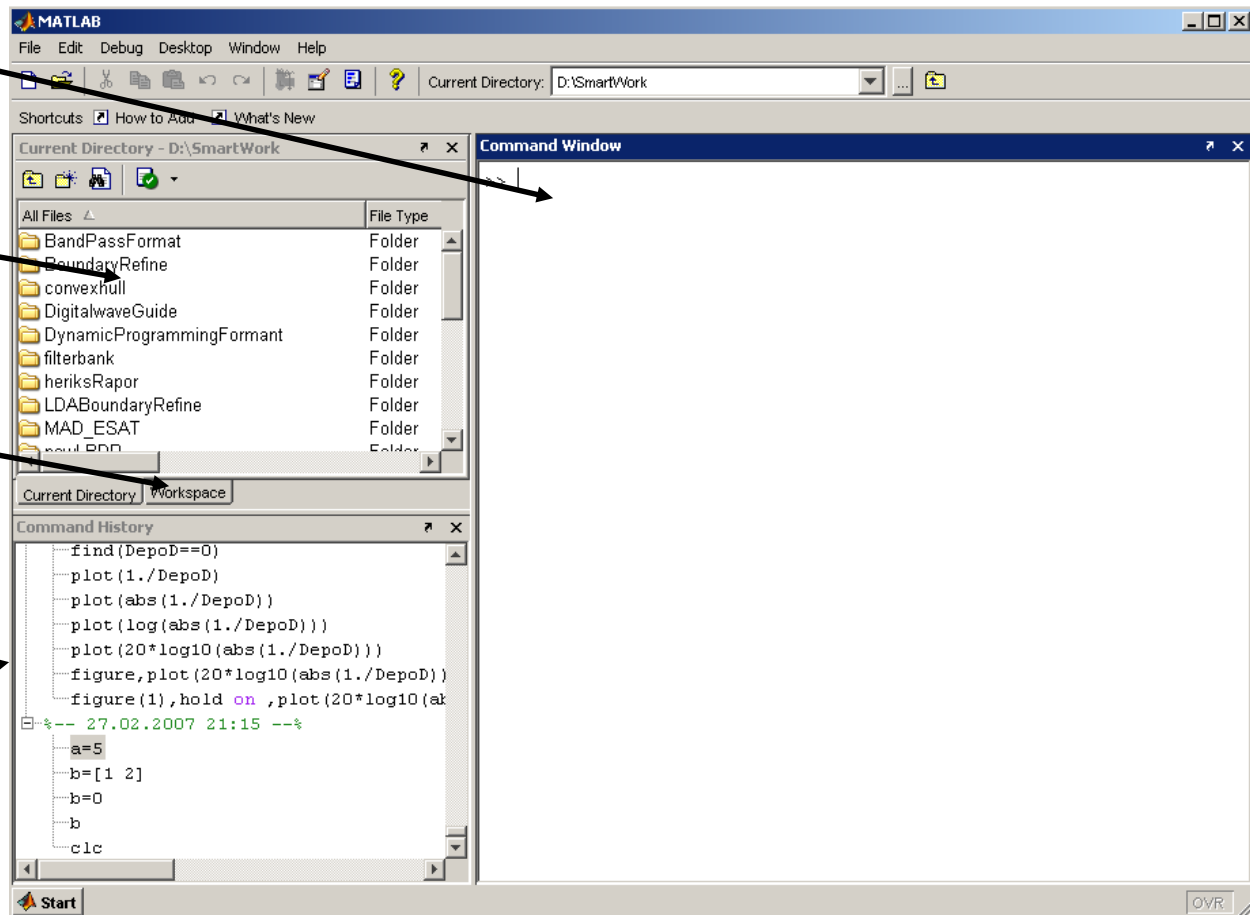
- Vizualizare foldere și fișiere
În format *.m (format Matlab)

Workspace

- vizualizare variabile
- Se pot vizualiza valoarea variabilelor prin Double click (se deschide fereastra Array Editor)

Command History

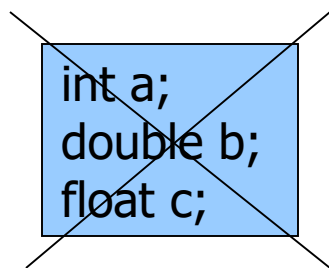
- Vizualizare istoric comenzi
- Salvarea unei sesiuni prin utilizarea unui jurnal de comenzi (diary)



I. Introducere în Matlab

Variable în Matlab

- În Matlab nu este nevoie să se inițializeze sau să se declare tipul variabilelor:



```
int a;  
double b;  
float c;
```

- Toate variabilele sunt create ca și matrici în format double

Exemplu:

```
>>x=5;  
>>x1=2;
```

- Conform exemplului anterior aceste variabile vor fi matrici de dimensiuni 1x1.

I. Introducere în Matlab

Structură Matlab

- Pentru a se vizualiza conținutul variabilelor trebuie doar sa se scrie numele variabilei în Command Window:

```
>> a
```

```
a =
```

```
12
```

```
>>
```

```
>> a*2
```

```
a =
```

```
24
```

```
>>
```

I. Introducere în Matlab

Structură Matlab

- Fereastra de Workspace reprezintă memoria curentă a Matlab-ului;
- Poate manipula variabilele stocate în workspace.

```
>> b=10;
```

```
>> c=a+b
```

```
c =
```

```
22
```

```
>>
```

I. Introducere în Matlab

Workspace Matlab

- **Comenzi de manipulare a Workspace-ului**

- *whos* – prezintă variabilele și dimensiunea acestora

<i>Name</i>	<i>Size</i>	<i>Bytes</i>	<i>Class</i>
<i>a</i>	<i>1x1</i>	<i>8</i>	<i>double array</i>
<i>b</i>	<i>1x1</i>	<i>8</i>	<i>double array</i>
<i>c</i>	<i>1x1</i>	<i>8</i>	<i>double array</i>

Grand total is 3 elements using 24 bytes

- *clear* – realizează ștergerea de variabile din workspace

```
>> clear a b; % delete a și b din workspace
```

```
>>
```

```
>> clear all; % șterge toate variabilele din workspace
```

```
>>
```

I. Introducere în Matlab

Operații matrici

- Nu este nevoie să se actualizeze / declare dimensiunile matricei:

```
>> A = [3 2 1; 5 1 0; 2 1 7]
```

```
A =
```

```
3    2    1
```

```
5    1    0
```

```
2    1    7
```

```
>>
```

Paranteze pătrate pentru a defini matricea

Punct și virgulă pentru a marca trecerea la o altă linie

I. Introducere în Matlab

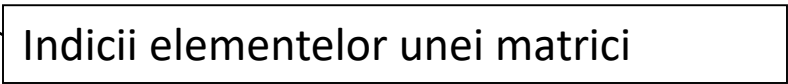
Accesarea valorilor unor matrici

- Accesarea elementelor unei matrici:

```
>> A(1,2)
```

```
ans=
```

```
2
```



Indicii elementelor unei matrici

A =

3 2 1

5 1 0

2 1 7

I. Introducere în Matlab

Alocare valori matrici în Matlab

- un vector $x = [1 \ 2 \ 5 \ 1]$

$x =$

1 2 5 1

- o matrice $x = [1 \ 2 \ 3; \ 5 \ 1 \ 4; \ 3 \ 2 \ -1]$

$x =$

1 2 3

5 1 4

3 2 -1

I. Introducere în Matlab

Operatorul „:”

- Foarte utilizat în Matlab;
- În traducere ar însemna “de la X la Y”

```
t = 1:10
```

```
t =
```

```
1 2 3 4 5 6 7 8 9 10
```

```
k = 2:-0.5:-1
```

```
k =
```

```
2 1.5 1 0.5 0 -0.5 -1
```

```
B = [1:4; 5:8]
```

```
x =
```

```
1 2 3 4  
5 6 7 8
```

I. Introducere în Matlab

Funcții definire funcții în Matlab

- `zeros(M,N)` - generează o matrice de valori de 0 de dimensiune MxN.

```
x = zeros(1,3)
x =
    0    0    0
```

- `ones(M,N)` - generează o matrice de valori de 1 de dimensiune MxN.

```
x = ones(1,3)
x =
    1    1    1
```

- `rand(M,N)` - generează o matrice de valori random distribuite uniform în intervalul (0,1) de dimensiune MxN.

```
x = rand(1,3)
x =
    0.9501    0.2311    0.6068
```


I. Introducere în Matlab

Indecșii matricilor

- Indecșii matricei încep de la 1 (nu de la 0 (ca în C))
- Indecșii matricei trebuie să fie numere întregi pozitive

```
A =  
  
     3     5     3  
     6     8     2  
     2     7     3
```

```
>> A(6)  
  
ans =  
  
     7
```

```
>> A(3,2)  
  
ans =  
  
     7
```

```
>> A(2,:)   
  
ans =  
  
     6     8     2
```

```
>> A(1:2,2)  
  
ans =  
  
     5  
     8
```

A(-2), A(0)

Error: ??? Subscript indices must either be real positive integers or logicals.

A(4,2)

Error: ??? Index exceeds matrix dimensions.

I. Introducere în Matlab

Concatenare matrici

- $x = [1 \ 2], \ y = [4 \ 5], \ z = [0 \ 0]$

$$A = [x \ y]$$

1 2 4 5

$$B = [x ; y]$$

1 2

4 5

$$C = [x \ y ; z]$$

Error:

??? Error using ==> vertcat CAT arguments dimensions are not consistent.

I. Introducere în Matlab

Operații matrici

+ sumă

- diferență

* multiplicare

/ împărțire

^ putere

' transpusă

I. Introducere în Matlab

Operații matrici

Fie A și B:

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

A =

1	2	3
4	5	6
7	8	9

```
>> B = [3 5 2; 5 2 8; 3 6 9]
```

B =

3	5	2
5	2	8
3	6	9

Sumă

```
>> X = A + B
```

X =

4	7	5
9	7	14
10	14	18

Diferență

```
>> Y = A - B
```

Y =

-2	-3	1
-1	3	-2
4	2	0

Produs

```
>> Z = A * B
```

Z =

22	27	45
55	66	102
88	105	159

Transpusă

```
>> T = A'
```

T =

1	4	7
2	5	8
3	6	9

I. Introducere în Matlab

Operatorul „.” (element cu element)

- .^{*} multiplicare element cu element
- .[/] împărțire element cu element
- .[^] ridicare la putere element cu element

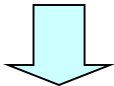
I. Introducere în Matlab

Operatorul „.” (element cu element)

```
A = [1 2 3; 5 1 4; 3 2 1]
```

A =

```
1 2 3
5 1 4
3 2 -1
```



```
x = A(1,:)
```

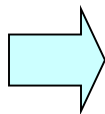
x =

```
1 2 3
```

```
y = A(3 ,:)
```

y =

```
3 4 -1
```



```
b = x .* y
```

b =

```
3 8 -3
```

```
c = x ./ y
```

c =

```
0.33 0.5 -3
```

```
d = x.^2
```

d =

```
1 4 9
```

```
K = x^2
```

Errorr:

??? Error using ==> mpower Matrix must be square.

```
B = x*y
```

Errorr:

??? Error using ==> mtimes Inner matrix dimensions must agree.

I. Introducere în Matlab

Comenzi de informații

- *help*

>> help whos % afișează documentație pentru funcția whos

>> lookfor convert % caută funcțiile care conțin termenul convert în prima linie a răspunsului comenzii help

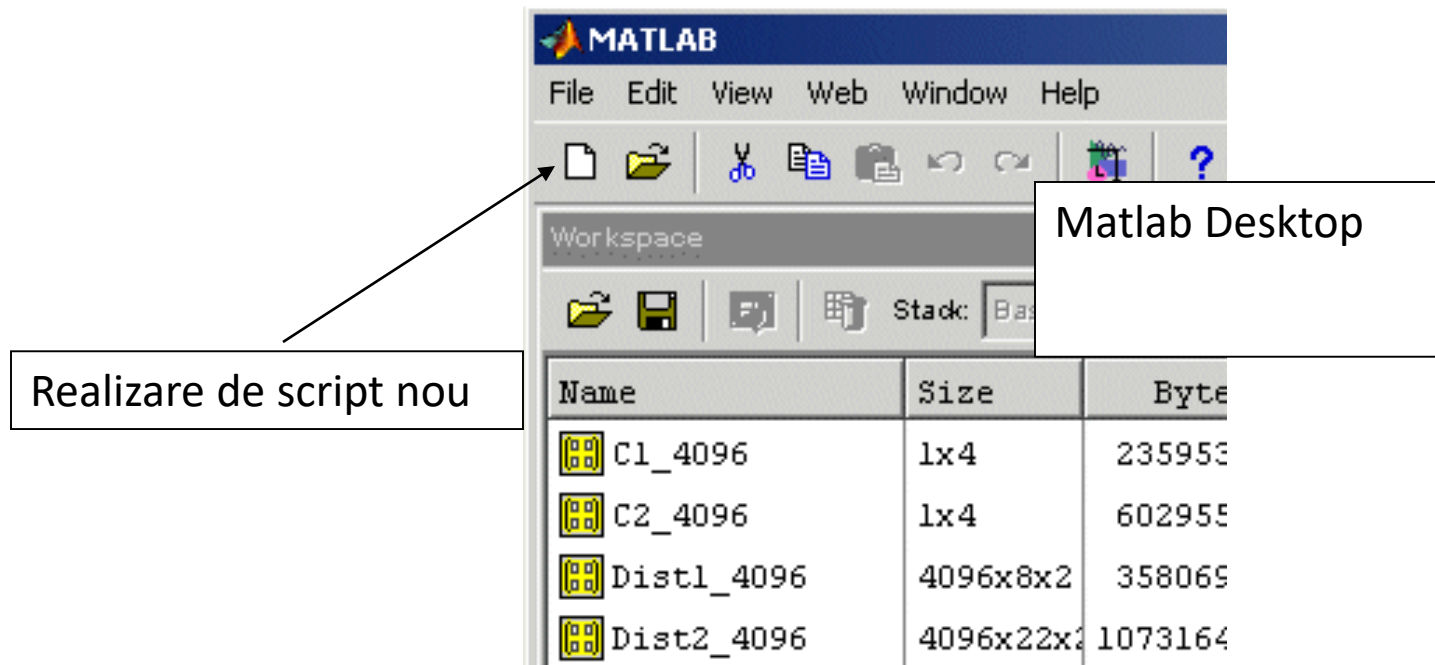
- Deschidere documentație Matlab

>> helpdesk

I. Introducere în Matlab

Realizare scripturi

- Mai multe comenzi matlab pot fi grupate în scripturi

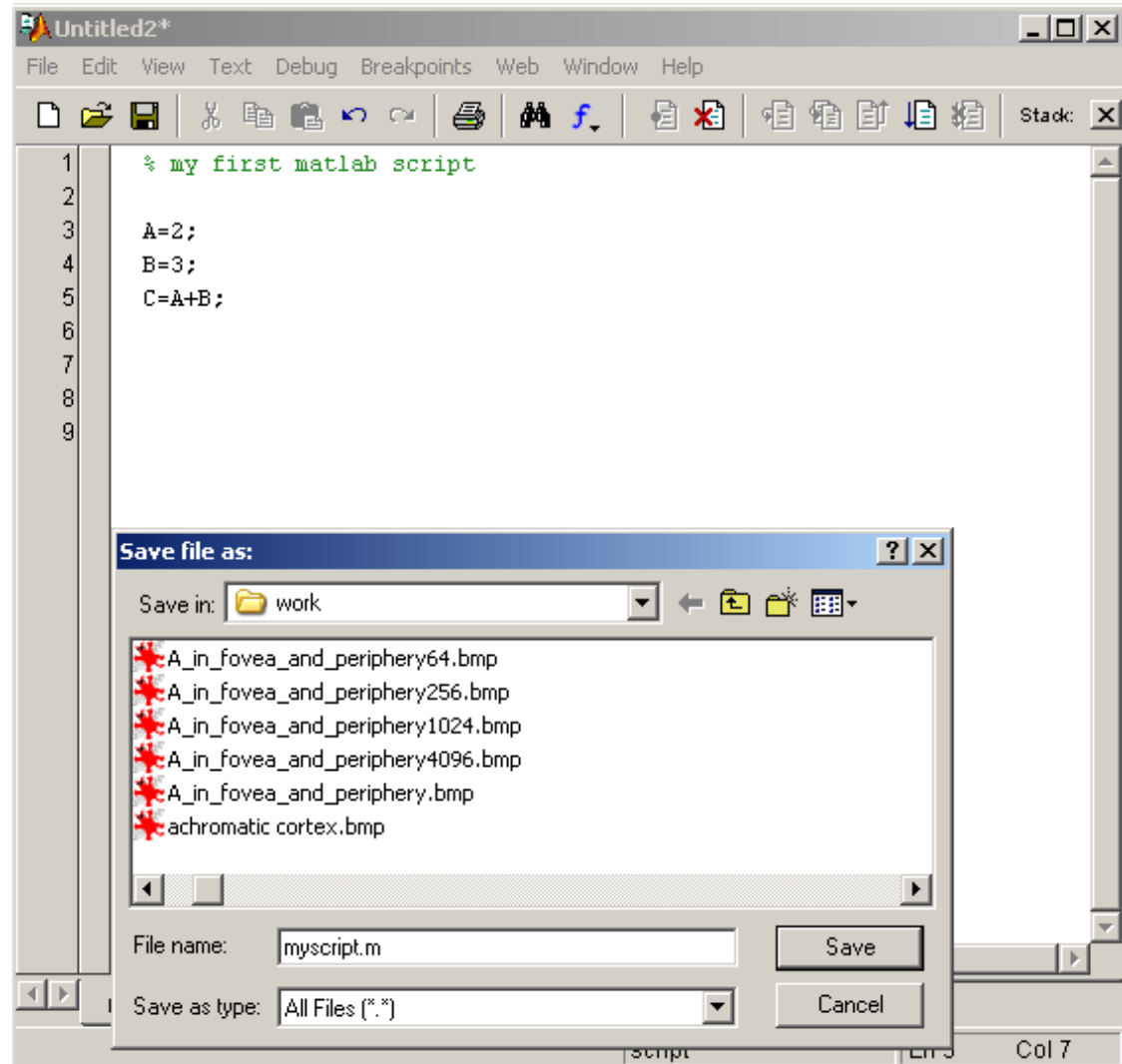


I. Introducere în Matlab

Realizare scripturi

- Scripturile vor manipula și stoca variabile în Matlab Workspace (memorie).
- Acestea pot fi chemate din linia de comandă Matlab prin scrierea fișierului scriptului.

>> myscript



I. Introducere în Matlab

Funcții în Matlab

- Programarea în Matlab.
- Userii pot scrie funcții care pot fi chemate din linia de comandă.
- Funcțiile pot accepta variabile de intrare și pot avea ca ieșire un set de variabile de ieșire.
- Funcțiile nu manipulează variabilele din cadrul Matlab Workspace.
- Numele fișierului care conține funcția trebuie să fie același cu cel al funcției
- Atenție la cazurile în care mai multe funcții au același nume.
- Funcțiile deschise pentru editare prin utilizare comenzii **open**.

I. Introducere în Matlab

Funcții în Matlab

```
>> l=iterate(5)
```

```
l =
```

```
1    4    9   16   25
```

Lista de
variabile de
ieșire

function keyword

Comentarii referitoare la
funcție

Bloc repetitiv for

Numele funcției

Variabile de
intrare

```
1 function O=iterate(n)
2 % iterate(n) outputs the square of the
3 % integers upto integer n
4
5
6 for i=1:n
7     O(i)=i*i;
8 end
```



Accesează comentariile
din funcția Matlab cu
funcția help
>> help iterate

I. Introducere în Matlab

Funcții în Matlab

```
>> [i j]=sort2(2,4)
```

```
i =
```

```
4
```

```
j =
```

```
2
```

```
>>
```

Bloc
condițional if

Funcțiile pot avea mai multe
variabile de ieșire.

```
1 function [p,q]=sort2(a,b)
2 % sort2(a,b) sorts a and b in descending
3 % order
4
5 if a>b,
6     p=a;
7     q=b;
8 else
9     p=b;
10    q=a;
11 end
```

I. Introducere în Matlab

Instrucțiuni condiționale

- Sintaxă:

```
if (Condition_1)
    instrucțiuni
elseif (Condition_2)
    instrucțiuni
elseif (Condition_3)
    instrucțiuni
else
    instrucțiuni
end
```

Exemple

```
if ((a>3) & (b==5))
    instrucțiuni;
end
```

```
if (a<3)
    instrucțiuni;
elseif (b~=5)
    instrucțiuni;
end
```

```
if (a<3)
    instrucțiuni;
else
    instrucțiuni;
end
```

I. Introducere în Matlab

Instrucțiuni repetitive

- Sintaxă instrucțiune for

```
for i=Index_Array  
    instrucțiuni  
end
```

Exemple

```
for i=1:100  
    instrucțiuni;  
end
```

```
for j=1:3:200  
    instrucțiuni;  
end
```

```
for m=13:-0.2:-21  
    Some Matlab Commands;  
end
```

```
for k=[0.1 0.3 -13 12 7 -9.3]  
    instrucțiuni;  
end
```

I. Introducere în Matlab

Instrucțiuni repetitive

- Sintaxă instrucțiune while

while (condition)

Matlab Commands

end

Exemplu

```
while ((a>3) & (b==5))  
    instrucțiuni;  
end
```

I. Introducere în Matlab

Debugging

- Atașare de breakpoints

```
>> [i j]=sort2(2,4)
```

```
K>>
```

```
K>> whos
```

Name	Size	Bytes	Class
a	1x1	8	double array
b	1x1	8	double array

Grand total is 2 elements using 16 bytes

```
K>> a
```

```
a =
```

```
2
```

```
K>> return
```

```
i =
```

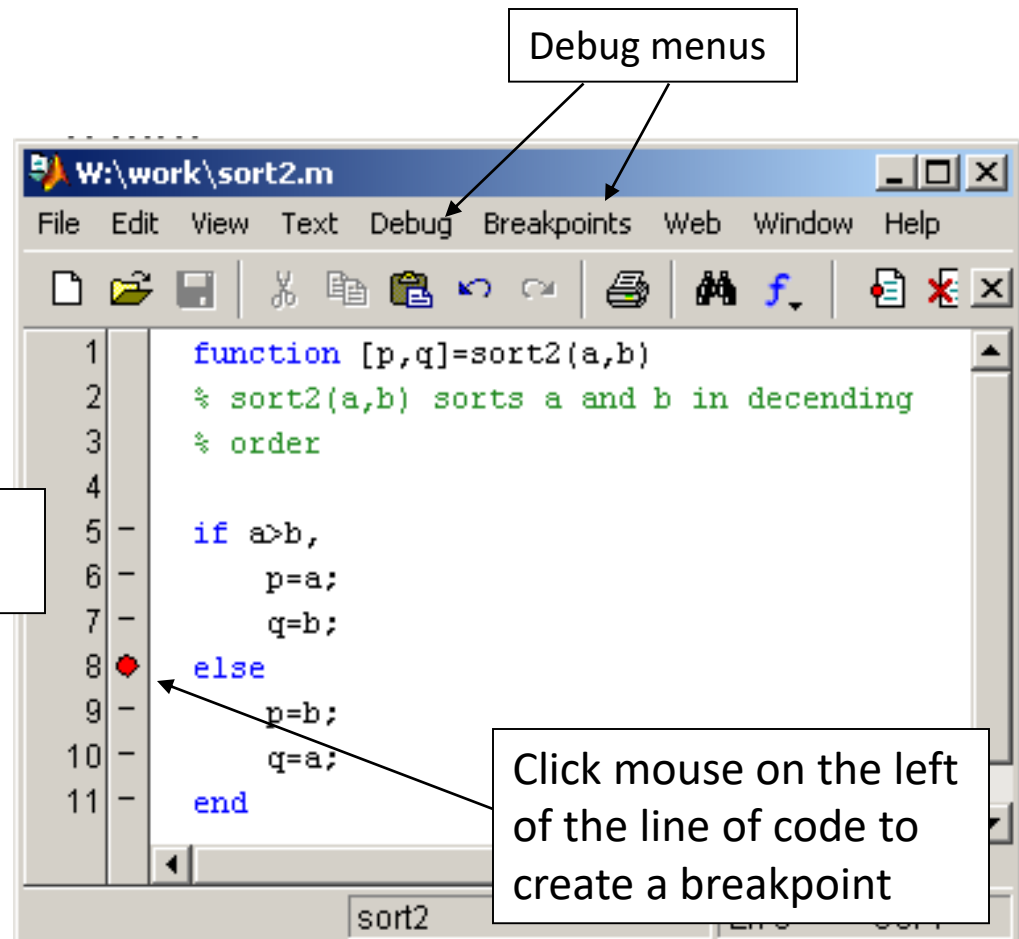
```
4
```

```
j =
```

```
2
```

local function
workspace

exit debug
mode



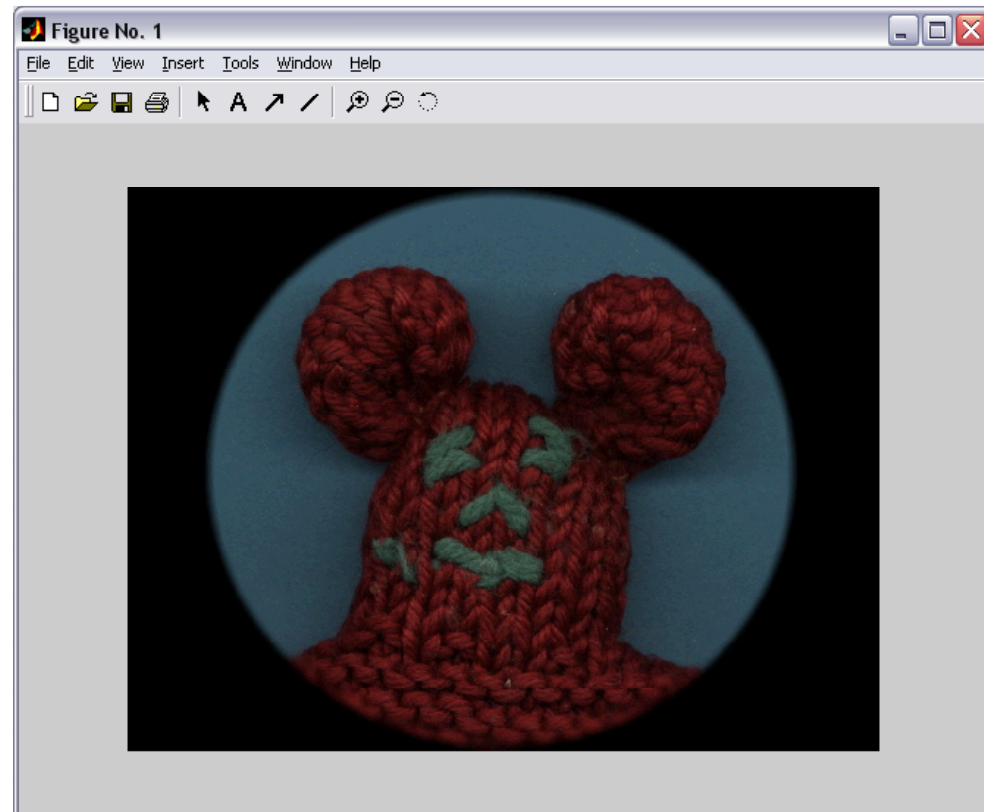
I. Introducere în Matlab

Prelucrare imagini

În Matlab imaginile color pot fi tratate ca niște matrici tri-dimensionale!
($M \times N \times 3$):

Încărcarea unei imagini:

```
a = imread('picture.jpg');  
% citire imagine  
imshow(a); %afișare imagine
```

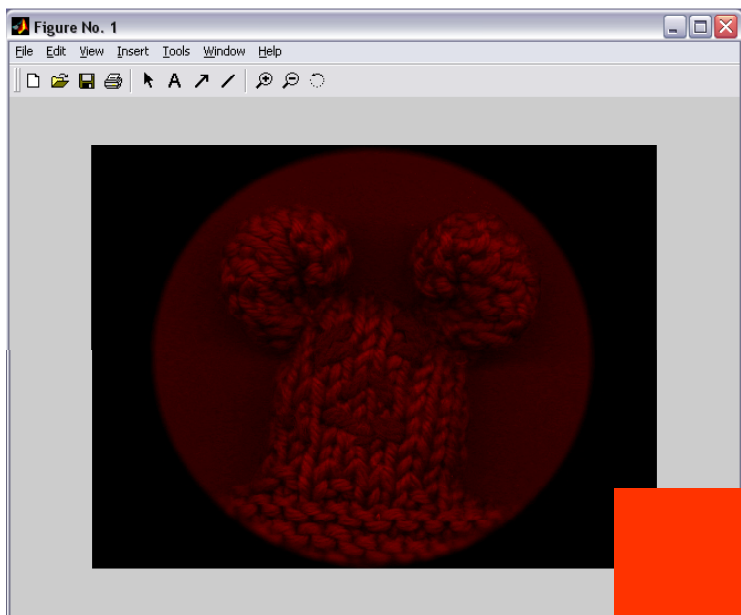


I. Introducere în Matlab

Prelucrare imagini

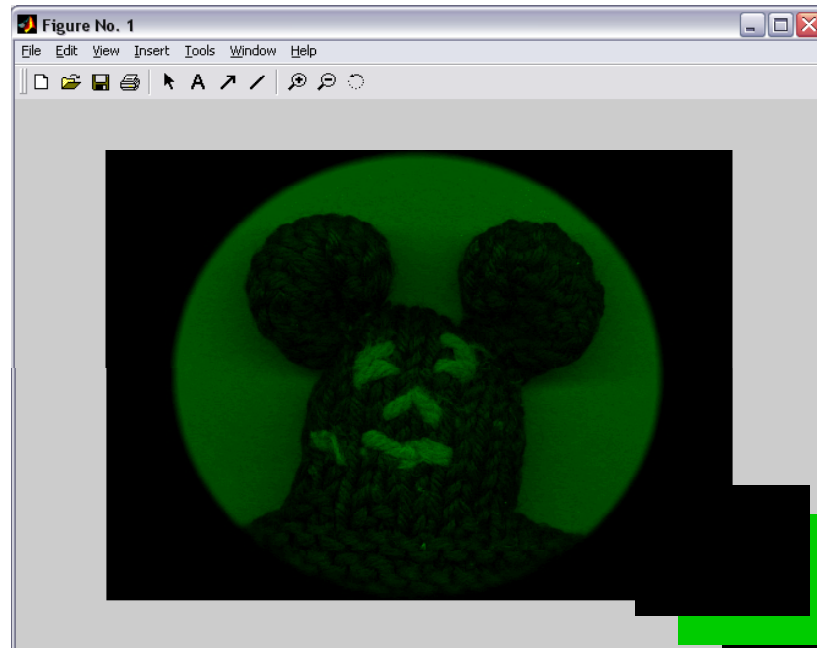
Afișarea planului roșu:

```
a(:,:,2:3) = 0;  
imshow(a);
```



Afișarea planului verde:

```
a(:,:, [1 3]) = 0;  
imshow(a);
```



I. Introducere în Matlab

Citire fișiere text

Funcția:

`[A,B,C,...] = textread(filename,format)`

Ex: pentru fișierul

```
1 2
3 4
```

utilizăm `[A B] = textread(filename,'%d %d')`
vom obține `A = [1;3]` și `B=[2;4]`

I. Introducere în Matlab

Citire fișiere csv

Funcția:

$[A] = \text{csvread}(\text{filename})$

Ex: pentru fișierul

1,2

3,4

utilizăm $[A] = \text{csvread}(\text{filename})$

vom obține $A = [1 \ 2; 3 \ 4]$

I. Introducere în Matlab

Scriere fișiere csv

Funcția:

csvwrite(filename, variabilă)

Ex: pentru fișierul

$A = [1 \ 2; 3 \ 4]$

csvwrite(filename,A)

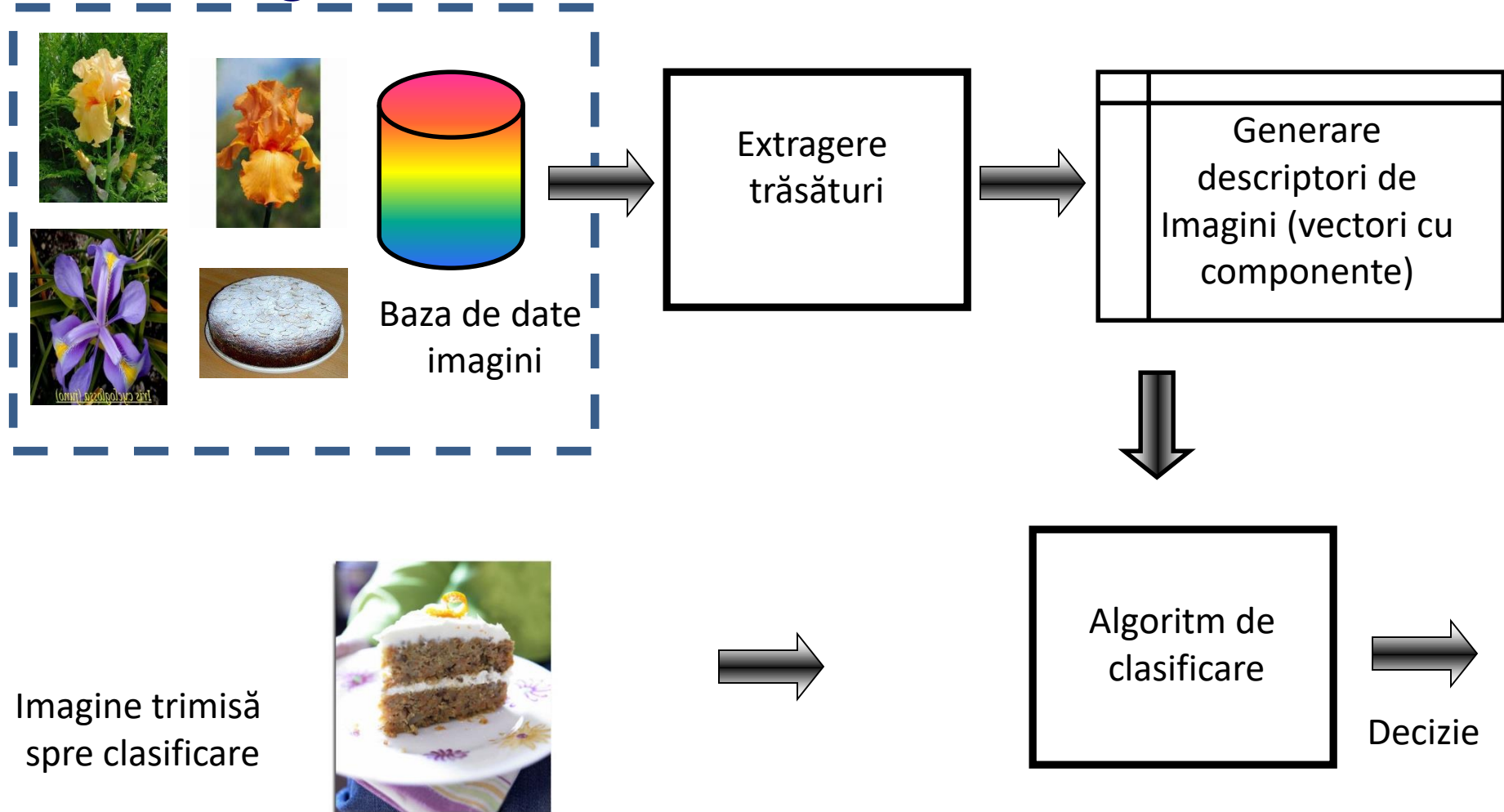
vom obține fișierul cu conținutul:

1,2

3,4

II. Clasificare de baze de date de imagini

Schemă generală

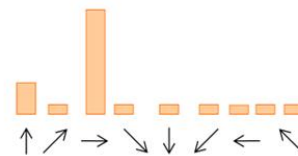


II. Clasificare de baze de date de imagini

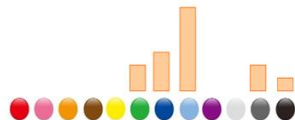
Descriptori de imagini



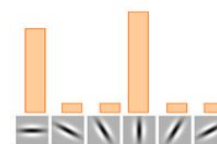
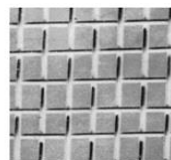
pixeli



muchii



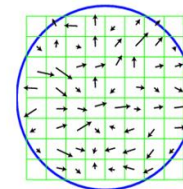
culoare



textură



formă



puncte de
interes

II. Clasificare de baze de date de imagini

Color moments (CM)

- Se împarte imaginea în 3x3 regiuni;
- Pentru fiecare celulă se calculează media, deviația standard și skewness pe o regiune a imaginii

$$E_i = \frac{1}{N} \sum_{j=1}^N p_{ij} \quad , \quad \sigma_i = \left(\frac{1}{N} \sum_{j=1}^N (p_{ij} - E_i)^2 \right)^{\frac{1}{2}} \quad \text{and} \quad s_i = \left(\frac{1}{N} \sum_{j=1}^N (p_{ij} - E_i)^3 \right)^{\frac{1}{3}}$$

- Se concatenează cei trei parametri pentru fiecare celulă (3x3x3 = 81 elemente).

Stricker M., Dimai A. Spectral Covariance and Fuzzy Regions for Image Indexing. Machine Vision and Applications, vol. 10., p. 66-73, 1997

II. Clasificare de baze de date de imagini

Color moments (CM)

- Pentru prelucrarea descriptorului se va utiliza funcția Matlab:

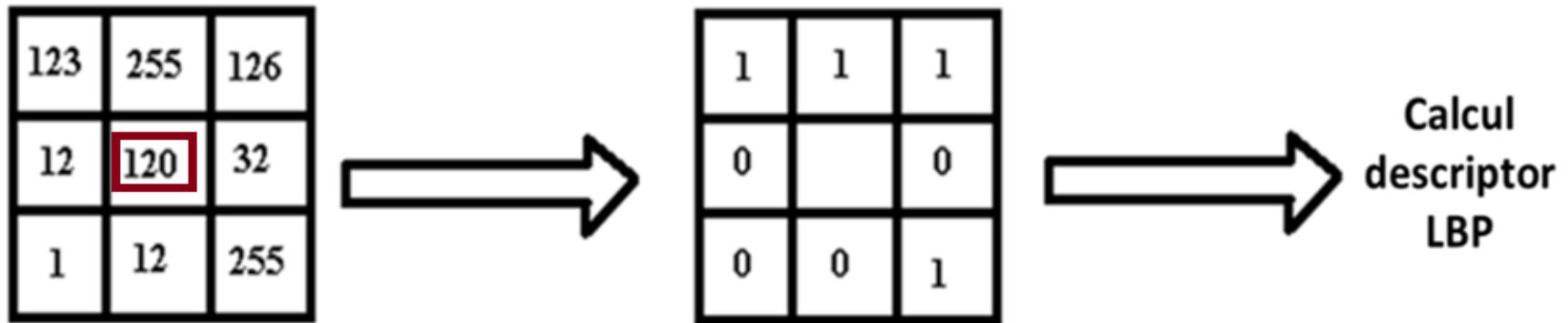
cm = extractCM(im);

unde *cm* reprezintă descriptorul rezultat iar *im* este o matrice de dimensiune (L x I x 3) asociată imaginii RGB.

II. Clasificare de baze de date de imagini

Locally Binary Patterns (LBP)

- Pentru fiecare pixel din imagine se aplică un prag egal cu valoarea punctului central din nucleul de căutare;
- Pentru fiecare pixel (x_c, y_c) se va calcula următorul parametru:



- Se creează o histogramă a valorilor generate (256 de combinații posibile);
- În cazul în care se repetă acest proces pe mai multe nuclee / scale ale imaginilor, descriptorul final va fi generat din concatenarea histogramelor generate

II. Clasificare de baze de date de imagini

Locally Binary Patterns (LBP)

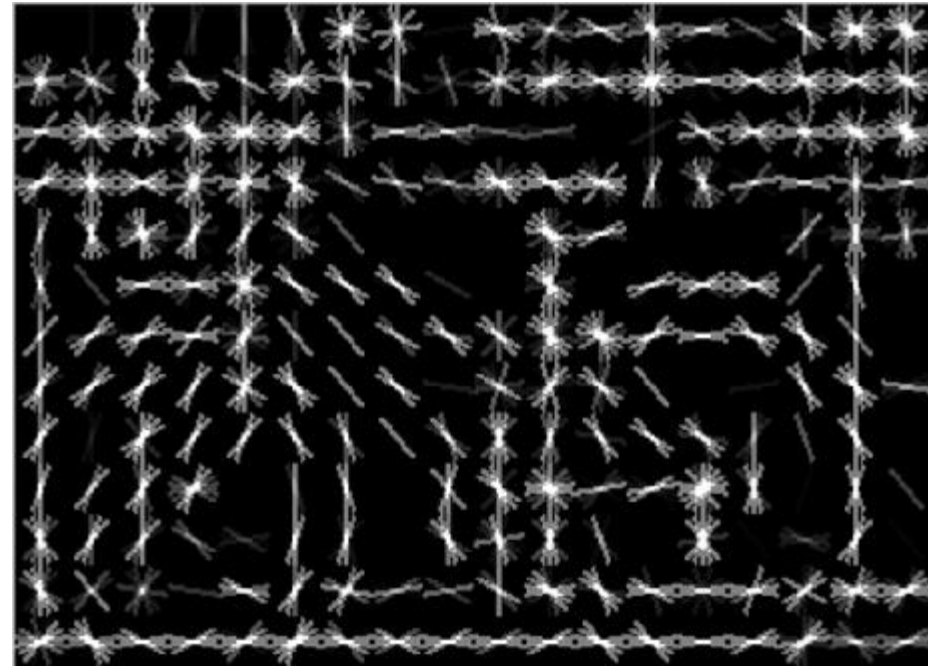
- Pentru prelucrarea descriptorului se va utiliza funcția Matlab:

lbp = extractLBP(im);

unde *lbp* reprezintă descriptorul rezultat iar *im* este o matrice de dimensiune $(L \times l \times 3)$ asociată imaginii.

II. Clasificare de baze de date de imagini

Histograms of Oriented Gradients (HoG)



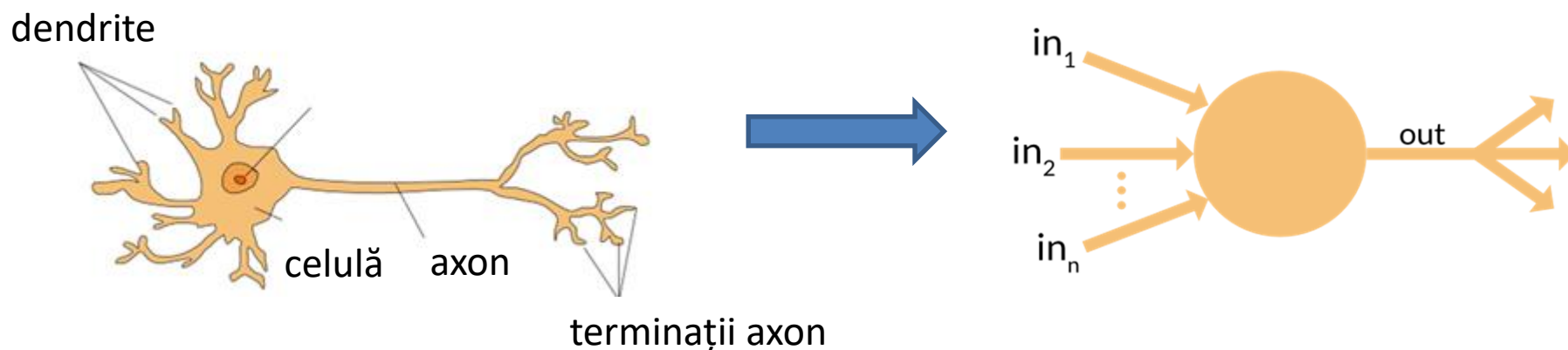
Pentru generarea descriptorului se va utiliza funcția:

hog = extractHoG(im);

III. Modele de Machine Learning

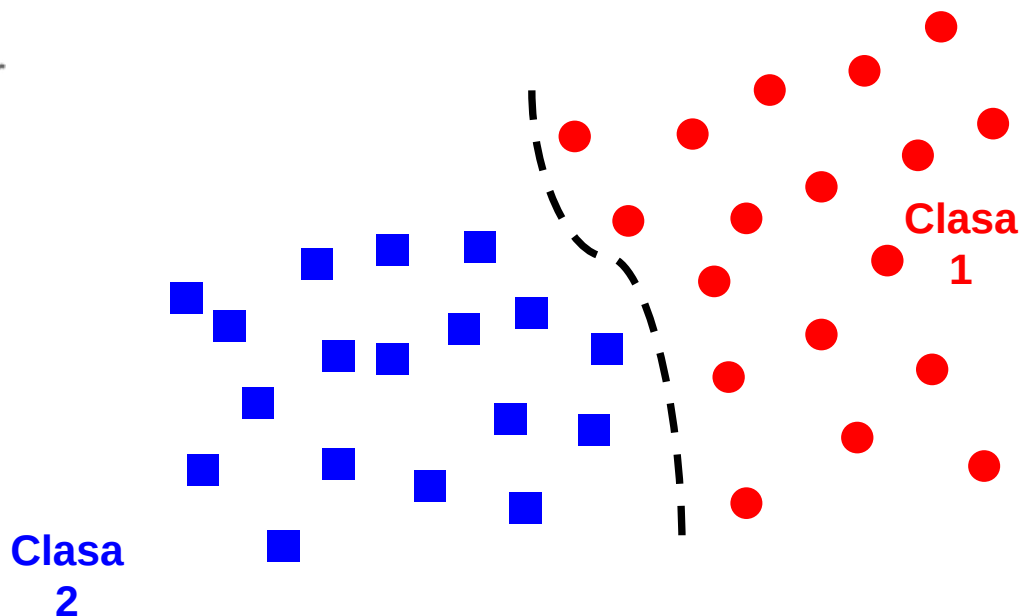
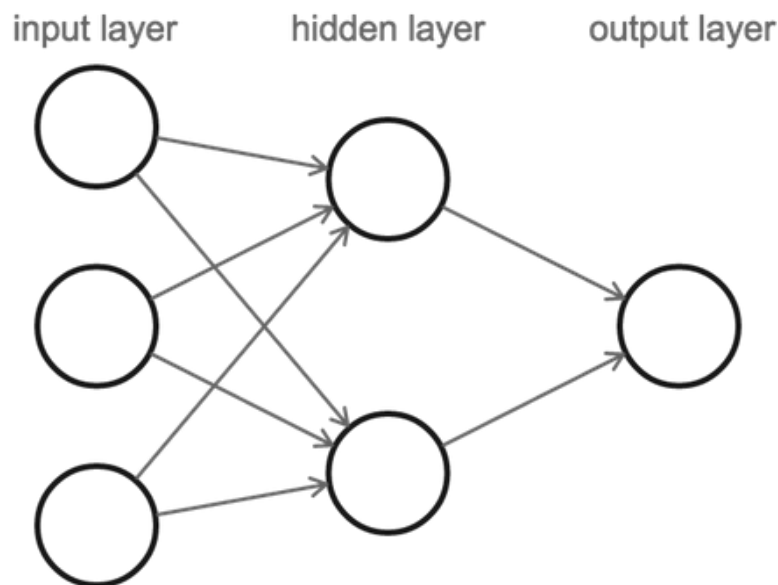
Rețele neuronale

- Rețelele neuronale reprezintă o mulțime de elemente de prelucrare neliniară care operează în paralel și care sunt legate între ele în structuri ce seamănă cu rețelele neuronale biologice.
- Model inspirat din rețelele neuronale din creierul uman.



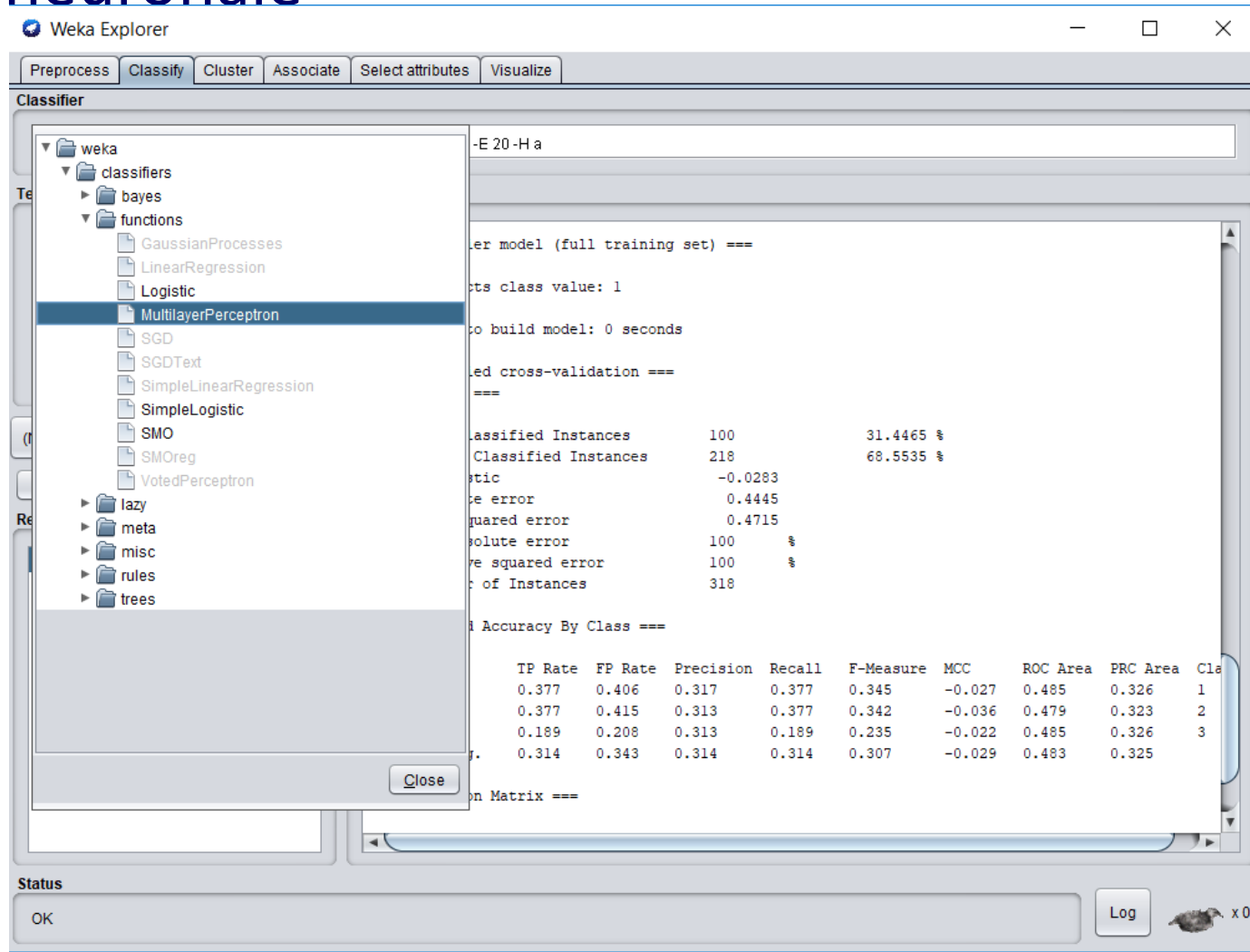
III. Modele de Machine Learning

Rețele neuronale



- Modele foarte bine adaptate pentru a găsi cea mai bună separație între două clase
- Structură complicată cu mulți parametri care nu sunt ușor de optimizat.
- Apare destul de des fenomenul de overfitting.

Rețele neuronale



III. Modele de Machine Learning

Parametrii rețelelor neuronale

weka.gui.GenericObjectEditor

weka.classifiers.functions.MultilayerPerceptron

About

A Classifier that uses backpropagation to classify instances. More Capabilities

GUI: False

autoBuild: True

batchSize: 100

debug: False

decay: False

doNotCheckCapabilities: False

hiddenLayers: 1

learningRate: 0.3

momentum: 0.2

nominalToBinaryFilter: True

normalizeAttributes: True

normalizeNumericClass: True

numDecimalPlaces: 2

reset: True

seed: 0

trainingTime: 500

validationSetSize: 0

validationThreshold: 20

Open... Save... OK Cancel

Afișare arhitectură rețea

Numărul de elemente dintr-un batch de antrenare

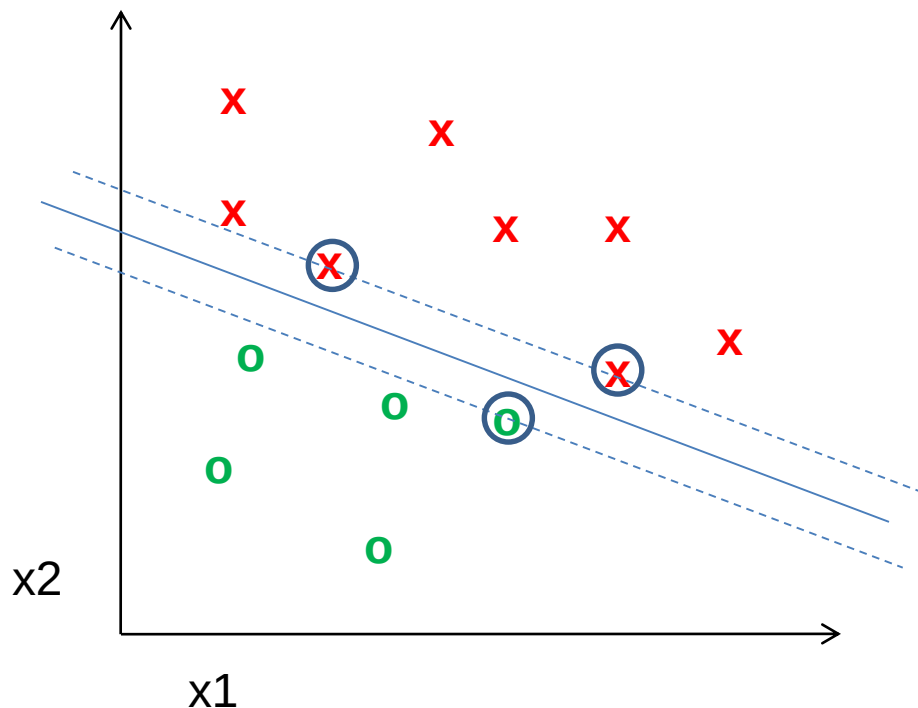
Valoarea ratei de învățare scade odată cu fiecare epocă

Numărul de straturi ascunse dintr-o rețea

Rata de învățare

V. Modele de Machine Learning

Support Vector Machines – model liniar

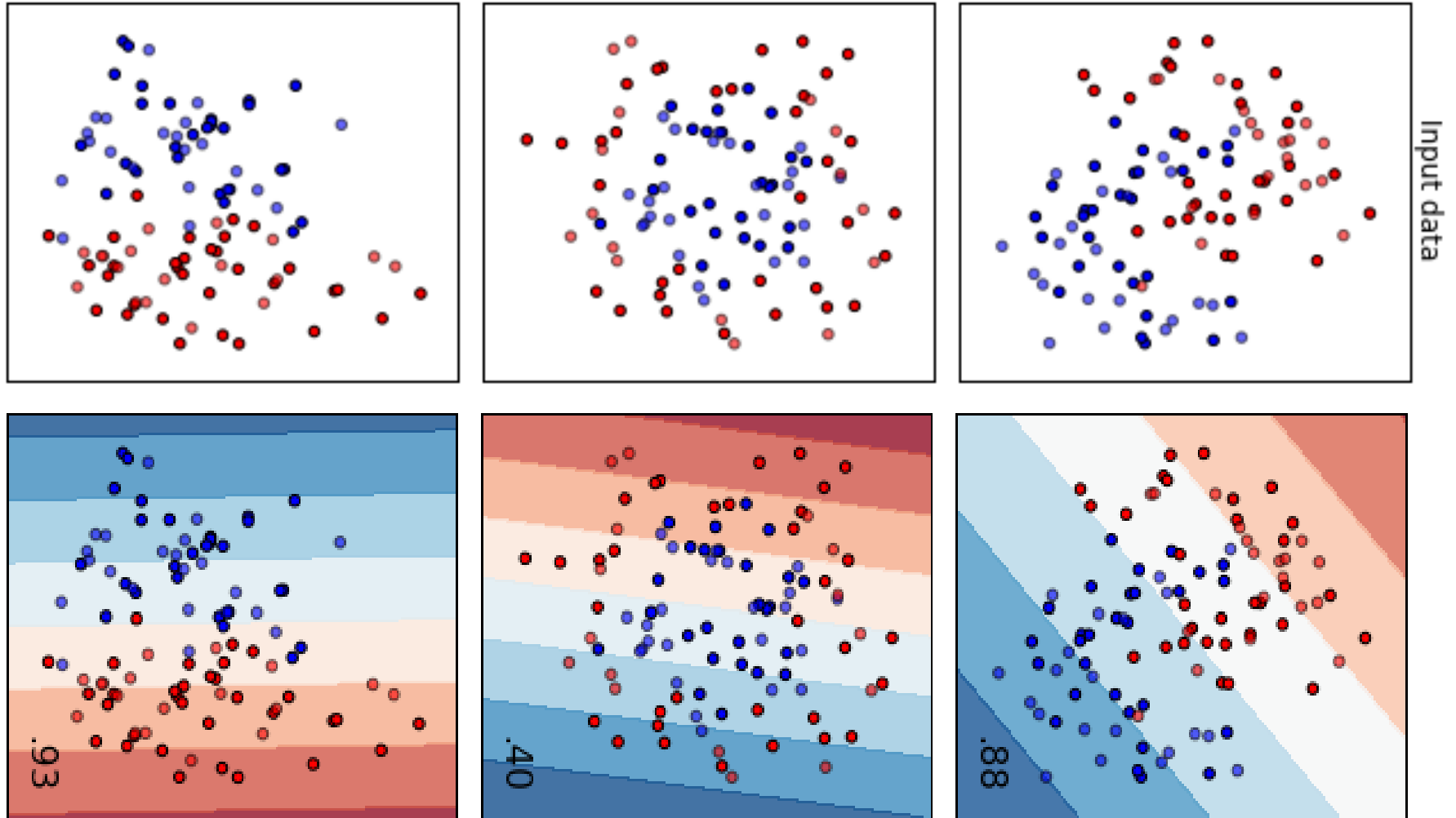


- Caută o funcție ce separă două clase în mod optim:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

III. Modele de Machine Learning

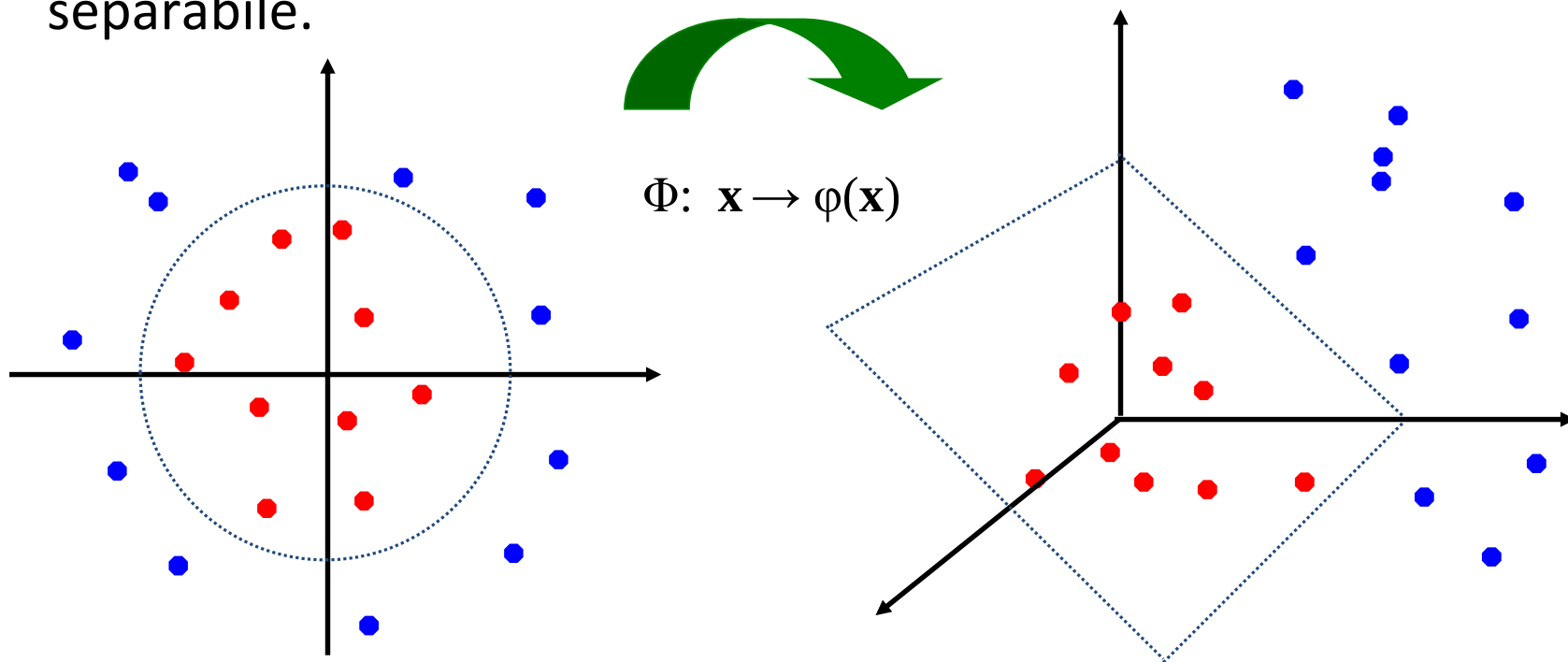
Support Vector Machines – model liniar



III. Modele de Machine Learning

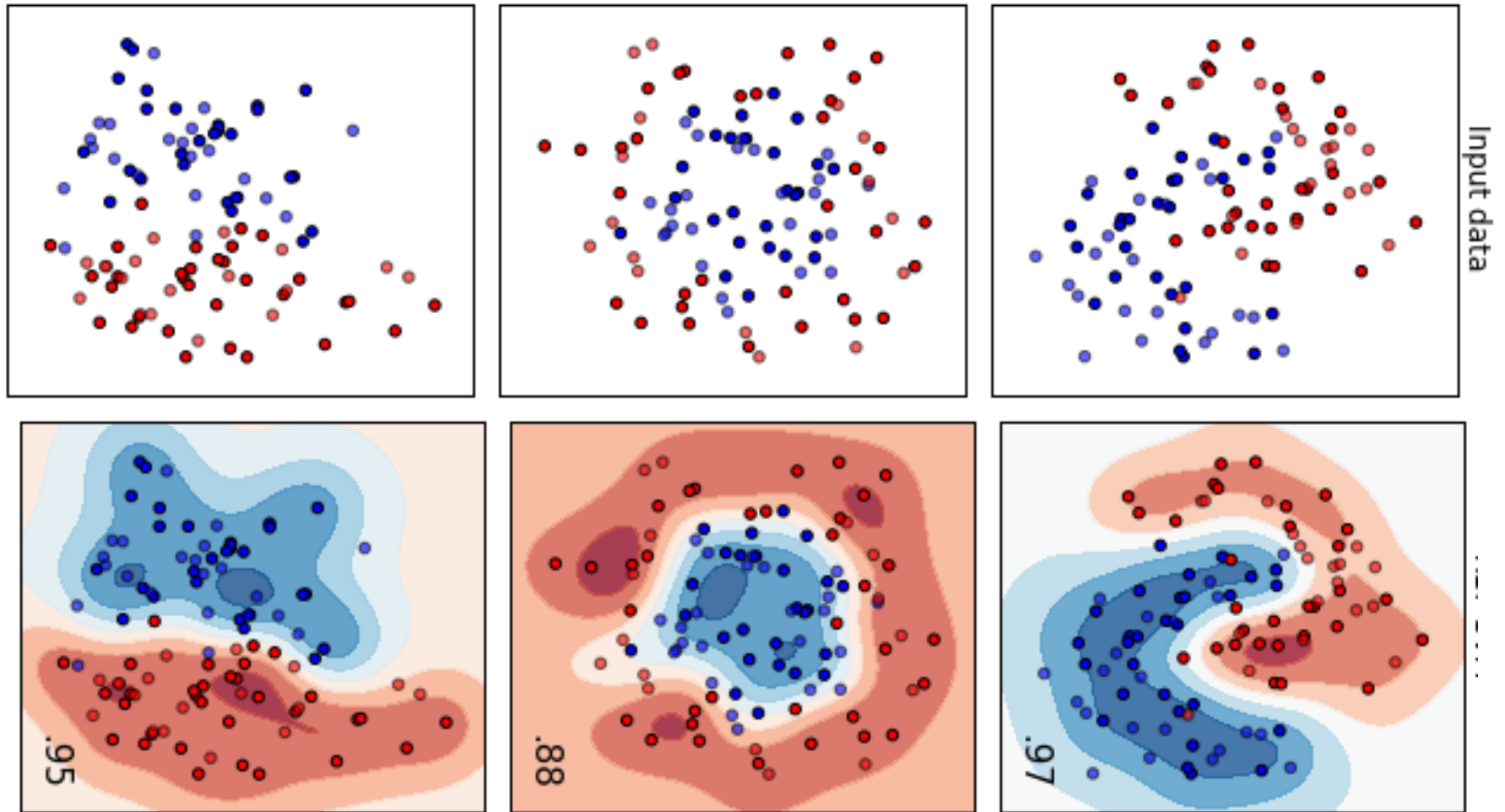
Support Vector Machines – model neliniar

- Idee de bază: spațiul inițial poate fi mapat către un spațiu multidimensional în care trăsăturile de antrenare devin liniar separabile.



III. Modele de Machine Learning

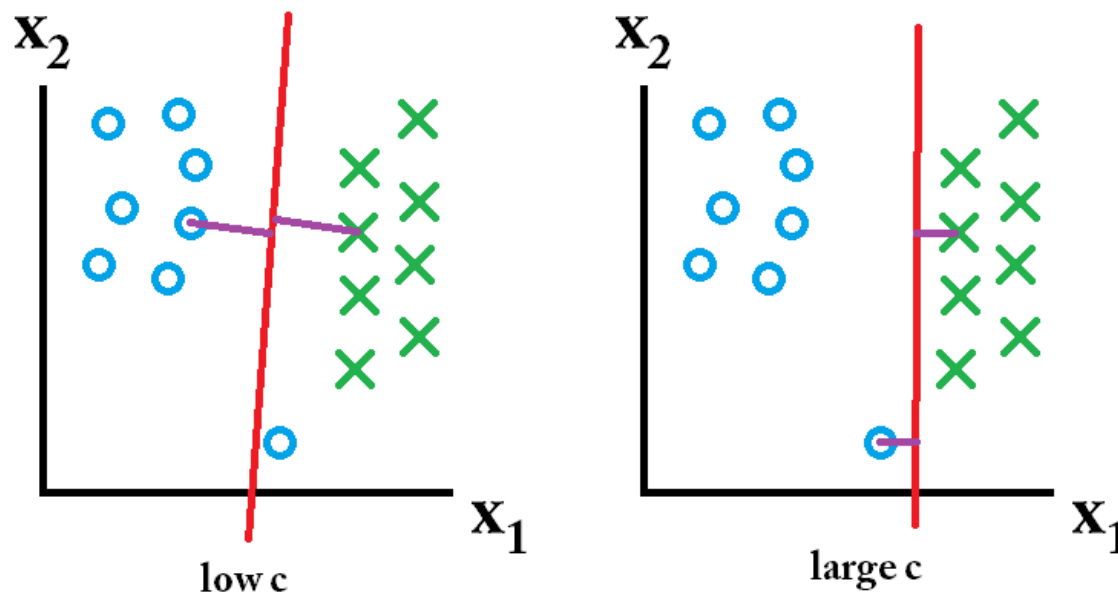
Support Vector Machines – model neliniar



III. Modele de Machine Learning

Optimizare SVM

- C
Modalitatea de calcul a hiperplanului de despărțire a claselor:
 - dacă modelul este prea adaptat pe date (C are o valoare mare) – modelul va suferi un proces de overfitting
 - dacă modelul va fi foarte simplu acesta nu va putea generaliza



- Gamma și tipul de nucleu utilizat

III. Modele de Machine Learning

Optimizare SVM

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic"

weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007" -calibrator "weka.classifiers.functions.Logistic"

Test model on test split: 0.01 seconds

====

Classified Instances	38	35.1852 %
Classified Instances	70	64.8148 %
Static	0	
Mean error	0.4444	
Mean squared error	0.4714	
Absolute error	99.8391 %	
Mean squared error	99.8002 %	
Number of Instances	108	

Accuracy By Class ==

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
1	0.000	0.000	?	0.000	?	?	0.452	0.272	1
2	0.000	0.000	?	0.000	?	?	0.313	0.301	2
3	1.000	1.000	0.352	1.000	0.521	?	0.551	0.361	3
g.	0.352	0.352	?	0.352	?	?	0.438	0.314	

Confusion Matrix ==

```
<-- classified as
a = 1
b = 2
c = 3
```

Status

OK

Log

x 0

III. Modele de Machine Learning

Optimizare SVM

The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The 'Classifier' list on the left has 'SMO' (Sequential Minimal Optimization) selected. The 'weka.classifiers.functions.SMO' configuration dialog is open, showing various parameters. Two red arrows point from text labels to specific settings:

- A red arrow points from the text 'Parametrul C' to the 'c' parameter, which is set to 1.0.
- A red arrow points from the text 'Tipul de nucleu' to the 'kernel' parameter, which is set to PolyKernel -E 1.0 -250007.

The dialog also includes an 'About' section, a 'batchSize' of 100, 'buildCalibrationModels' set to False, a 'calibrator' set to 'SGD -F 0 -L 0.01 -R 1.0E-4 -E 500 -C 0.001 -S 1', 'checksTurnedOff' set to False, 'debug' set to False, 'doNotCheckCapabilities' set to False, 'epsilon' set to 1.0E-12, 'filterType' set to 'Normalize training data', 'numDecimalPlaces' set to 2, 'numFolds' set to -1, 'randomSeed' set to 1, and 'toleranceParameter' set to 0.001.

IV. Clasificare de baze de date de imagini

Baze de date experimente laborator

Experimentele se vor efectua pe două baze de date:

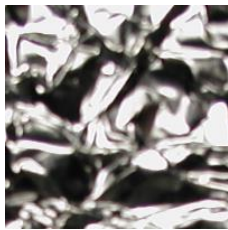
- Bază de date de texturi (parte a bazei de texturi KTH – „*Textures under varying Illumination, Pose and Scale*”)
 - conține 4 clase (81 de imagini pe clasă)
- Bază de date de imagini naturale (conține 3 clase preluate din baza *Caltech 101*: prăjituri / iriși și monede) -> 3 clase (106 imagini pe clasă).

<http://www.nada.kth.se/cvap/databases/kth-tips/>

http://www.vision.caltech.edu/Image_Datasets/Caltech101/

IV. Clasificare de baze de date de imagini

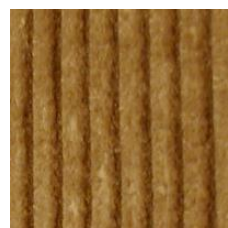
Texturi



Clasa *aluminiu*



Clasa *pâine*



Clasa *țesătură*



Clasa *bumbac*

Sursă imagini <http://www.nada.kth.se/cvap/databases/kth-tips/>

IV. Clasificare de baze de date de imagini

Imagini naturale



Clasa cake



Piece of 1730's Spanish Coin



Obverse

Reverse



Clasa coins



Clasa iris

Sursă imagini http://www.vision.caltech.edu/Image_Datasets/Caltech101/

V. Exerciții

Pentru cele două baze de date (texturi și imagini naturale):

- Realizați un program în Matlab care calculează rând pe rând cei trei descriptori (Color Moments, LBP și HoG):
 - Salvați descriptorii în fișiere csv;
 - Transformați aceste fișiere în format arff.
- Schemă algoritm

Citește fișier de configurare (train.txt) (imagePaths, labels)

descriptors %inițializare spațiu pentru export în fișier arff

pentru fiecare imagine din bază (contor i)

- im = citeșteImagine(i)

- desc = CalculDescriptor(im);

- Adaugă variabilei descriptors -» concat (desc, labels(i))

Salvează variabila descriptors în fișier

V. Exerciții

- Importați rând pe rând aceste fișiere în Weka.
- Utilizați ca și clasificatori: ZeroR, arbori de decizie (J48), Nearest Neighbor (IBK), Naive Bayes, Support Vector Machines și rețele neuronale.
- Verificați care este cea mai bună pereche clasificator / descriptor.

IV. Exerciții

- Optimizați modelul SVM (încercați mai multe valori ale lui C și mai multe tipuri de nuclee)
- Optimizați modelul de rețea neuronală utilizat

Spor!