

Creazione di un classificatore PR-Tools

I classificatori PR-Tools sono degli oggetti che incapsulano i parametri e tutte le informazioni necessarie alla classificazione di un dataset tramite un opportuno algoritmo.

La creazione di un classificatore PR-Tools consiste nella creazione di una o più funzioni in grado di gestire tre possibili casi d'uso:

1. Creazione di un classificatore non addestrato
es: `W=ldc([],1)`
2. Addestramento di un classificatore
es: `W=ldc(ds); ds*W`
3. Classificazione di un dataset

```
function W = myClassifierPRT(ds,par1, par2, ...)
```

Nota:
Tipicamente le operazioni di Creazione e addestramento di un classificazione sono gestite da un'unica funzione.

La struttura tipica di una funzione che definisce un classificatore PR-Tools è la seguente:

```
function W = myClassifierPRT(ds,par1, par2, ...)
    % Verifica input
    if (nargin < 3)
        par2 = 0;
    end

    if (nargin < 2)
        par1 = 0;
    end

    %-(1)-----
    % Creazione classificatore non addestrato
    %-----
    if (nargin < 1) | (isempty(ds))
        % Creo l'oggetto classificatore non addestrato
        % tramite la funzione mapping
        W = mapping(mfilename,{par1,par2});
        W = setname(W,name);
        return
    end
```

```

%-(2)-----
% Addestramento Classificatore
%-----
% In fase di classificazione par1 è un oggetto
% di tipo "mapping"

if ~isa(par1, 'mapping')
    [numPatt numFeat] = size(ds);
    labslist = getlabslist(ds);
    numClass = length(labslist);

    %Calcolo parametri classificatore
    data.val1 = ...
    data.val2 = ...

    % Creo l'oggetto classificatore addestrato
    % tramite la funzione mapping
    W = mapping(mfilename, 'trained', data, ...
               labslist, numFeat, numClass);
    % Memorizzo il nome del classificatore
    W = setname(W, name);
end

```

```

%-(3)-----
% Classificazione
%-----
% In fase di classificazione par1 è un oggetto
% di tipo "mapping"

if isa(par1, 'mapping')
    w = par1;
    [numPatt, numFeatures]=size(ds);

    % Utilizzo i parametri stimati per calcolare le
    % probabilità a posteriori (o gli score)
    P_post = ....

    labslist = getlab(w);

    % Salvo l'oggetto dataset
    % memorizzando le probabilità a posteriori
    % al posto delle features
    W = setdata(ds, P_post, labslist);
end
return;

```

Esercizio: implementare un classificatore lineare binario

Modello Gaussiano: caso $\Sigma_i = \sigma^2 I$

$$g_i(x) = w_i^T x + w_{i0}$$

$$w_i = \frac{1}{\sigma^2} \mu_i; \quad w_{i0} = -\frac{1}{2\sigma^2} \mu_i^T \mu_i + \ln(P(\omega_i))$$

```
function W = linearDiscriminant(ds, par1);  
%  
% par1 = 0 : (default) la matrice di covarianza è stimata  
%           senza tenere conto della numerosità delle classi  
%  
% par1 = 1 : la matrice di covarianza è stimata tenendo  
%           conto della numerosità delle classi
```

Traccia:

- Utilizzare le funzioni “getlab(ds)” e “getdata(ds)” per estrarre labels e features
- Utilizzare le funzioni “mean(...)” e “cov(...)” di matlab per il calcolo dei parametri

```

function W = linearDiscriminant(ds, par1);
% par1 = 0 : (default) la matrice di covarianza è stimata
%          senza tenere conto della numerosità delle classi
%
% par1 = 1 : la matrice di covarianza è stimata tenendo
%          conto della numerosità delle classi

if (nargin < 2)
    par1 = 0;
end

name = 'liner discriminant';
%-(1)-----
% Creazione classificatore non addestrato
%-----
if (nargin < 1) | (isempty(ds))
    % Creo l'oggetto classificatore non addestrato
    % tramite la funzione mapping
    W = mapping(mfilename,{par1});

    % Memorizzo il nome del classificatore
    W = setname(W,name);
    return
end

```

```

if ~isdataset(ds)
    %ds non Ã un dataset
    error('ds non Ã un dataset PR-Tools');
end

%-(2)-----
% Addestramento Classificatore
%-----
% In fase di classificazione parl Ã un oggetto
% di tipo 'mapping'

if ~isa(par1, 'mapping')

    [numPatt numFeat] = size(ds);
    labslist = getlablist(ds);
    numClass = length(labslist);

    %Calcolo parametri classificatore
    labels = getlab(ds);

    dataTrn = getdata(ds);
    dataTrnA = dataTrn(labels==1,:);
    dataTrnB = dataTrn(labels==2,:);

```

```

    numPattA = size(dataTrnA,1);
    numPattB = size(dataTrnB,1);

% stimo le probabilit  a priori
data.P_A = numPattA/numPatt;
data.P_B = numPattB/numPatt;

% Calcolo i vettori media
data.m_A= mean(dataTrnA);
data.m_B= mean(dataTrnB);

% Calcolo la Varianza.
cov_A=cov(dataTrnA);
cov_B=cov(dataTrnB);

if par1 == 0,
    %varianza media
    data.varianza= (trace(cov_A) +trace(cov_B) )/4;
else
    %varianza pesata
    data.varianza=(numPattA*trace(cov_A) + ...
        numPattB*trace(cov_B) )/(2*numPatt);
end

```

```

% Creo l'oggetto classificatore addestrato
% tramite la funzione mapping
W = mapping(mfilename, 'trained', data, ...
            labslist, numFeat, numClass);
% Memorizzo il nome del classificatore
W = setname(W, name);
end

%-(3)-----
% Classificazione
%-----
% In fase di classificazione parl e' un oggetto
% di tipo "mapping"
if isa(parl, 'mapping')
    w = parl;

    [numPatt, numFeatures]=size(ds);
    labslist = getlab(w);
    numClass = length(labslist);

    %G1
    wA=(1/w.data.varianza) * w.data.m_A;
    wA0= -1/(2*w.data.varianza) * ...
        (w.data.m_A)*(w.data.m_A)' + log(w.data.P_A);

```

```

%G2
wB=(1/w.data.varianza) * w.data.m_B;
wB0= -1/(2*w.data.varianza) * ...
      (w.data.m_B)*(w.data.m_B)' + log(w.data.P_B);

```

```

% classificazione
dataTst = getdata(ds);
scoreG = zeros(numPatt, numClass);
for idx=1:numPatt
    x=dataTst(idx,:);
    %f(x)=gA(x)-gB(x);
    scoreG(idx,1) = wA*(x')+wA0;
    scoreG(idx,2) = wB*(x')+wB0;
end

```

```

% Salvo l'oggetto dataset
% memorizzando le probabilit  a posteriori
% al posto delle features
W = setdata(ds, scoreG, labslist);

```

```

end

```

```

return;

```