

INTRODUZIONE AI PRTOOLS

- Creazione dataset
- Grafici
- Estrazione etichette e features dai dataset
- Classificazione e calcolo dell'errore

I PRTools sono un pacchetto aggiuntivo, gratuito per uso accademico, che implementa una vasta serie di algoritmi per il PR e la manipolazione dei dataset

<http://www.prtools.org/>

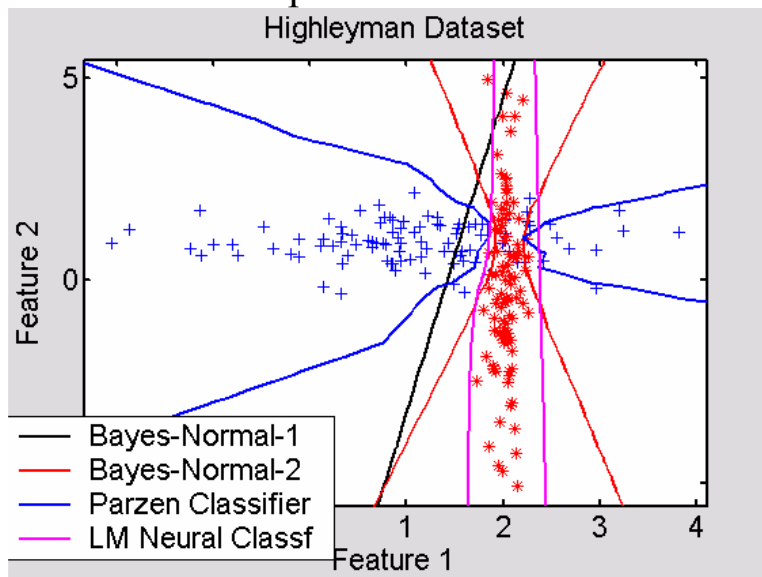
Permettono di:

- ‘incapsulare’ i pattern in oggetti dataset che ne rendono comoda la gestione
- utilizzare algoritmi già implementati
- implementare e gestire in modo uniforme nuovi algoritmi
- visualizzare graficamente i risultati

...

Per l'utilizzo è sufficiente inserire la cartella prtools nel path di matlab

Sono disponibili alcuni script che illustrano le caratteristiche del pacchetto. Ad esempio, `prex_plotc` crea un dataset e quattro classificatori, e mostra i risultati della classificazione. E' possibile visualizzare il dataset e le superfici di decisione dei classificatori.



I dataset nei prtools

Creazione del dataset:

- **A=dataset(datamatrix,labels,featlist,prob,lablist)**

- datamatrix: matrice con i dati per riga (ogni riga di feature = un pattern)
- labels: lista dei “target” di classe associate a ciascun pattern di datamatrix
- featlist: i “nomi” di ciascuna feature (es. [‘x’; ‘y’; ‘z’])
- prob: i “prior” di ciascuna classe: sum(prob)=1
- lablist: i “nomi” di ciascuna classe

Non è sempre necessario precisare tutti i parametri in dataset.

Esempio:

- **A=dataset([0 0; 0 1; 1 0; 1 1], [1 2 2 1]’)**

ESTRARRE I DATI DAL DATASET

-[nlab,lablist,m,k,c,prob,featlist]=dataset(A)

- per estrarre informazioni dal dataset, dove m, k, c sono rispettivamente il numero di pattern, la dimensione dello spazio delle feature, il numero di classi

E' possibile accedere ai campi dell'oggetto mediante i metodi preposti

GETDATA, GETLABELS,
GETFEATLAB, GETFEATSIZE, GETIDENT, , GETLABLIST,
GETLABTYPE, GETNAME, GETNLAB, GETOBJSIZE,
GETPRIOR, GETCOST, GETSIZE, GETTARGETS,
GETTARGETS, GETUSER, GETVERSION.

E' possibile accedere direttamente ai campi dell'oggetto dopo averlo trasformato in struttura

A.DATA = data

A.NLAB = numeric labels, index in lablist

A.OBJECTSIZE = number of objects or vector with its shape

A.FEATSIZE= number of features or vector with its shape

A.FEATLAB = feature labels

A.LABLIST = labels of the classes

A.FEATDOM = feature domains

A.PRIOR = prior probabilities

A.COST = classification cost matrix

A.TARGETS = dataset with soft labels or targets

A.LABTYPE = label type: 'crisp','soft' or 'target'

A.IDENT = identifier for objects (integer)

A.VERSION = PRTools version used for creating dataset

A.NAME = string with name of the dataset

A.USER = user field

Visualizzazione dei dati

Plot, subplot, figure, hold on / hold off

- funzioni base di visualizzazione di Matlab (usare help per i dettagli)

scatterd(dataset)

- proietta i pattern nello spazio delle feature

plotd(W)

- proietta la superficie di decisione descritta dal classificatore W nello spazio delle feature

ESEMPIO

Creare un 'dataset' con pattern della classe 1 uniformemente distribuiti in $[0,1] \times [0,1]$ e pattern della classe 2 uniformemente distribuiti in $[2,3] \times [2,3]$

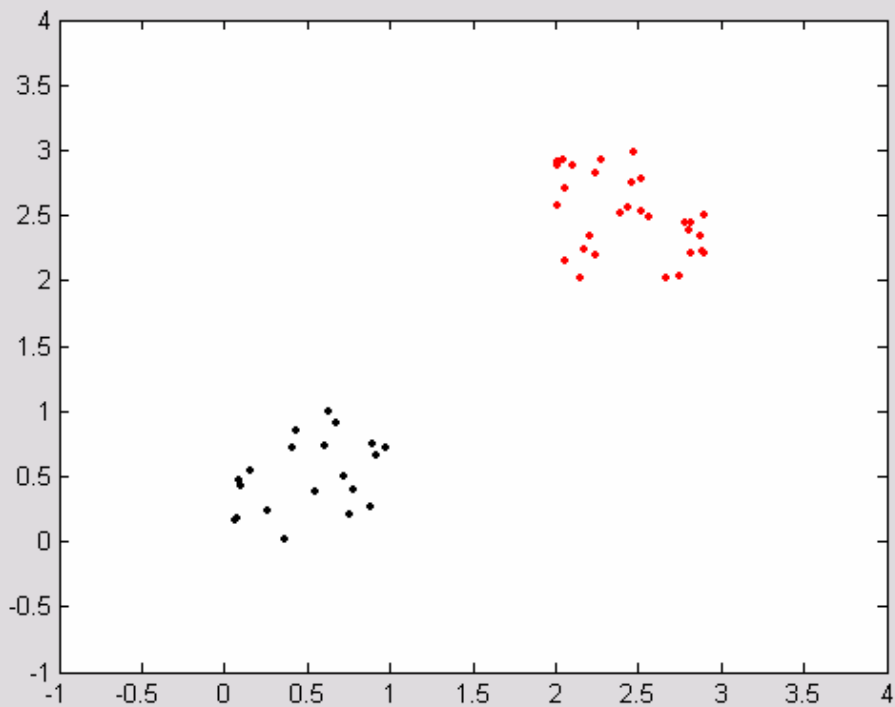
Memorizzate le features nella matrice *dati* e le etichette nella matrice *label*

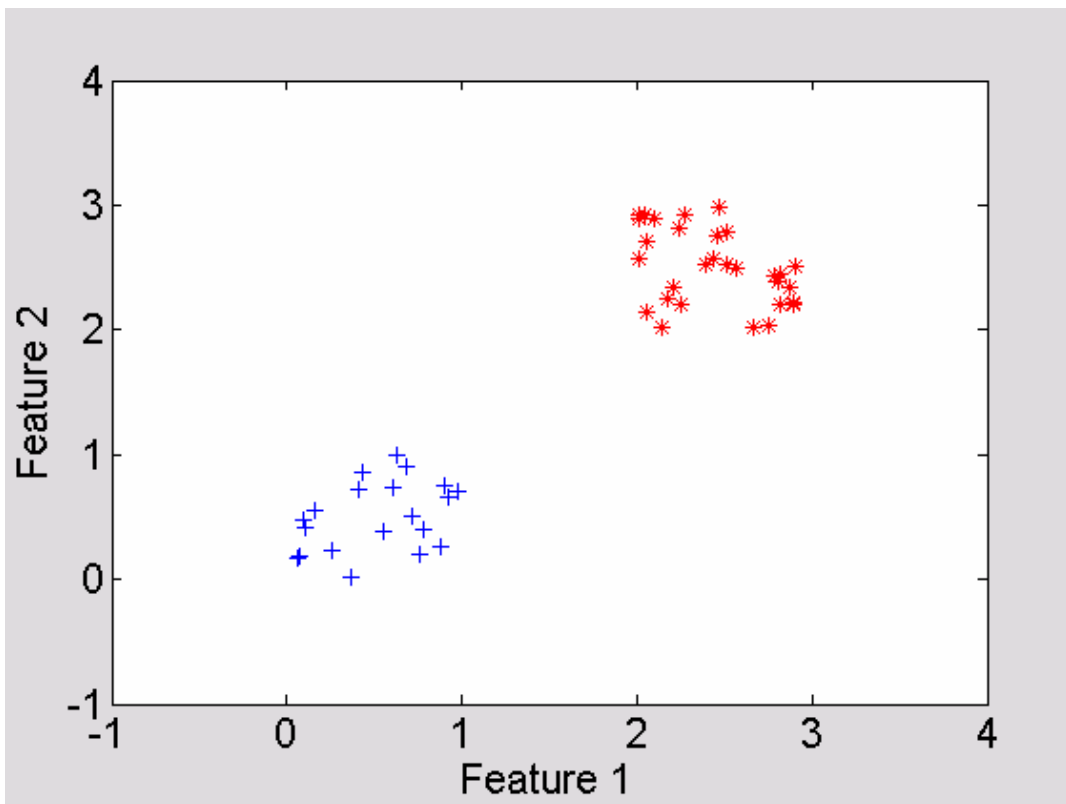
Visualizzate i dati mediante la funzione
`plot(dati_x, dati_y, parametri)`

Incapsulate i dati utilizzando la struttura dei prtools.

Visualizzate il risultato e verificate l'uguaglianza dei dati 'incapsulati' con i dati originali

- non eseguite i comandi direttamente, ma mediante uno script
- salvate tutti gli oggetti utilizzati
- Funzioni utili: `rand`, `dataset`, `struct`, `plot`, `scatterd`
(soluzione: `plotDataset.m`)





Ho incapsulato i dati in un oggetto dataset.

Per recuperarli posso utilizzare le funzioni `get...` oppure posso trasformare in struttura

```
mieiDati= struct (mio_randDataset)
all(all(mieiDati.data==dati))
all(mieiDati.nlab==label)
```

Dataset Artificiali a disposizione nei prtools

Sono presenti alcune routine per la generazione di dataset artificiali

gauss	multivariate Gaussian distributed data
gencirc	one-class circular dataset
gendat	classes from given data set
gendatb	banana shaped classes
gendatc	circular classes
gendatd	two difficult classes
gendath	Higleyman classes
gendatk	Nearest neighbour data generation
gendatl	Lithuanian classes
gendatm	8 2d classes
gendatp	Parzen density data generation
gendats	two Gaussian distributed classes

help gendatb

GENDATB Generation of banana shaped classes

A = GENDATB(N,S)

INPUT

N number of generated samples of vector with number of samples per class

S variance of the normal distribution (opt, def: s=1)

OUTPUT

A generated dataset

DESCRIPTION

Generation of a 2-dimensional 2-class dataset A of N objects with a banana shaped distribution. The data is uniformly distributed along the bananas and is superimposed with a normal distribution with standard deviation S in all directions. Class priors are $P(1) = P(2) = 0.5$.

Defaults: N = [50,50], S = 1.

ESEMPIO DI UTILIZZO

```
banana = gendatb(400)
```

```
Banana Set, 400 by 2 dataset with 2 classes:  
[201 199]
```

```
scatterd (banana) ;
```

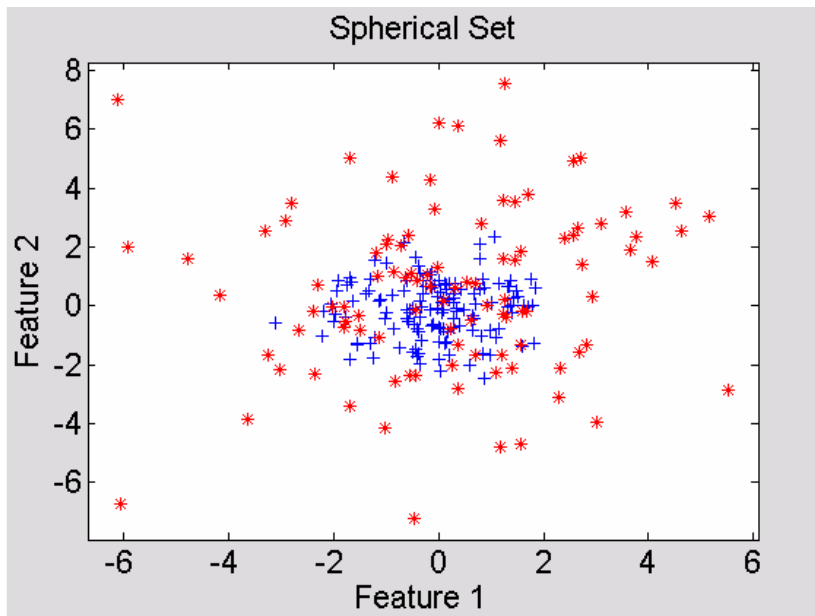
```
help gendatc
```

```
...
```

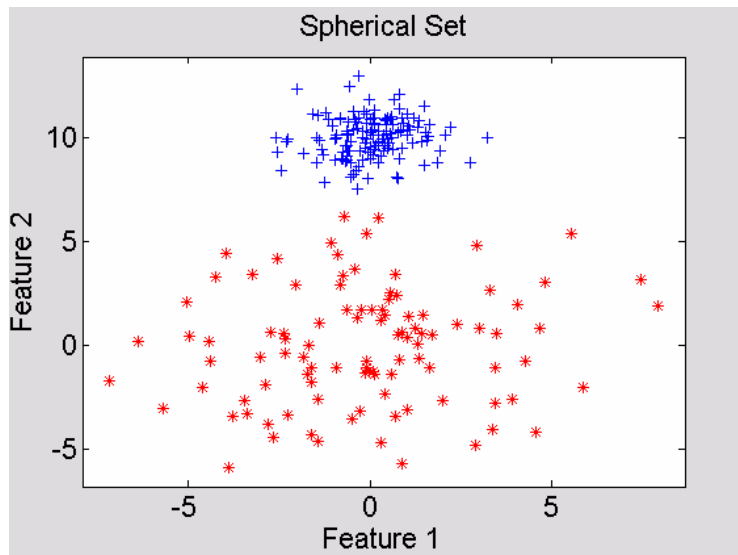
```
sferal=gendatc([140,100],2)
```

```
Spherical Set, 240 by 2 dataset with 2  
classes: [140 100]
```

```
scatterd(sferal)
```




```
sfera2=gendatc([140,100],2,[0, 10])  
scatterd(sfera2)
```



ESEMPIO

Esaminare i dati relativi al dataset ‘phoneme’ presenti sul sito del corso.

Convertirlo nel formato Prtools.

Visualizzarlo proiettandolo nelle varie dimensioni

Normalizzazione dataset

Consideriamo il dataset PHONEME originale, se analizziamo la matrice delle features (`features_set1`), tramite le funzioni `max()` e `min()`, possiamo vedere che il dataset non è normalizzato.

Esistono diversi modi per normalizzare le feature di un dataset

- le features devono avere valori tra 0 e 1
- le features devono avere media nulla e deviazione standard unitaria
- ...

Normalizzazione tra 0 e 1

Sia $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ l' i -esimo pattern.

Voglio ottenere:

$$x'_{ij} = \frac{x_{ij} - x_j^{(\min)}}{x_j^{(\max)} - x_j^{(\min)}} \in [0,1] ,$$

$$\begin{cases} x_j^{(\max)} = \max_{i=1,2,\dots,N} x_{ij} \\ x_j^{(\min)} = \min_{i=1,2,\dots,N} x_{ij} \end{cases}$$

(i è l'indice del pattern, j è l'indice della *feature*)

Normalizzazione con media nulla e dev. standard unitaria

Sia $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ l' i -esimo pattern.

Voglio ottenere:

$$x'_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}, \left\{ \begin{array}{l} \mu_j = \frac{1}{N} \sum_{i=1}^N x_{ij} \\ \sigma_j^2 = \frac{1}{N} \sum_{i=1}^N (x_{ij} - \mu_j)^2 \end{array} \right.$$

Nota:

In Matlab sono presenti due funzioni, **mean()** e **std()**, per il calcolo della media e della deviazione standard.

ESEMPIO: normalizz.m

Divisione random di un dataset

Spesso è utile dividere un dataset, in modo casuale, in varie parti:

- Dati artificiali, vogliamo creare un train e un test. Il test set rappresenta i pattern di test ‘non noti’; il training set rappresenta i pattern di cui è nota la classe di appartenenza
- Vogliamo usare solo una parte dei pattern per l'addestramento, e l'altra per la stima delle prestazioni

...

I PRTools mettono a disposizione una funzione specifica per eseguire questa operazione:

help gendat

GENDAT Random generation of datasets for training and testing

```
[A,B,IA,IB] = gendat(X,n)
```

Selects at random $n(i)$ vectors out of class i in the dataset X and stores them in A . The remaining vectors are stored in B . If n is a scalar, then n objects are selected for each class. By $n < 1$ relative sizes may be defined with respect to the original class sizes.

IA and IB are the indices of the objects selected from X for A and B .

Esempio:

```
[ds1, ds2] = gendat(phoneme_ds1, 0.3);
```

Classificatori PRTools

I PRTools mettono a disposizione vari classificatori in grado di **assegnare le etichette** e stimare le **probabilità a posteriori** di appartenenza alla classe.

ldc	Normal densities based linear classifier
qdc	Normal densities based quadratic classifier
fisherc	Minimum least square linear classifier
knnc	k-nearest neighbour classifier
lmnc	Feed forward neural network by Levenberg-Marquardt rule
...	...

Nota: dato che MatLab possiede un vasto toolbox di reti neurali, i PRTools non sviluppano questo aspetto, ma forniscono una semplice interfaccia verso il toolbox matlab.

Esaminiamo il processo di classificazione utilizzando un semplice classificatore lineare; il procedimento è analogo per tutti gli altri classificatori PRTools.

help ldc

```
LDC Linear Bayes Normal Classifier
(BayesNormal_1)

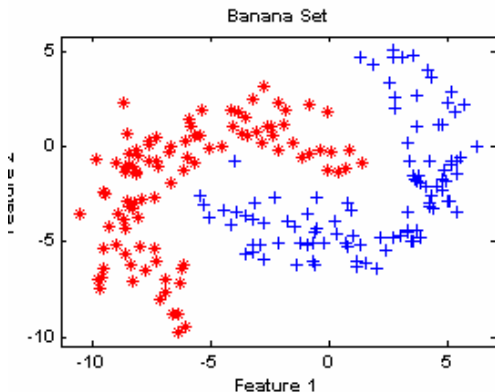
W = LDC(A,R,S)

INPUT
  A      Dataset
  R,S    Regularization parameters, 0 <= R,S
<= 1
        (optional; default: no
regularization, i.e. R,S = 0)
OUTPUT
  W      Linear Bayes Normal classifier
```

Tutti i classificatori presentano dei parametri aggiuntivi che permettono di regolarne l'addestramento. In questo caso i due parametri R ed S permettono di agire sul metodo con cui vengono stimati i parametri (media e matrice di covarianza).

Vediamo come effettuare la classificazione del dataset sintetico 'banana' con questo semplice classificatore lineare

- Generiamo il dataset sintetico
`ds_banana= gendatb(200,1) ;`
- Visualizziamo graficamente il Dataset creato
`scatterd(ds_banana) ;`



- Creiamo il classificatore

```
W = ldc(ds_banana);
```

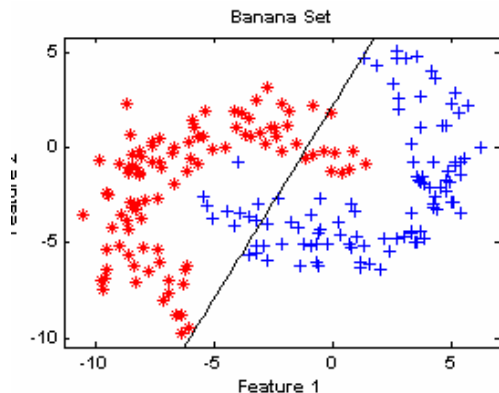
- Stimiamo le probabilità a posteriori

```
probPost = +classc(ds_banana*W);
```

- Visualizziamo la superficie di decisione

```
hold on;
```

```
plotc(W);
```



Calcolo accuratezza

L'accuratezza fornisce la percentuale di pattern correttamente classificati.

- Stimiamo le etichette di classe (label) con il nostro classificatore

```
labelAssegnate = labeld(ds_banana*W);
```

- Estraiamo le etichette vere dal dataset

```
labelVere = getlab( ds_banana );
```

- Confrontiamo etichette vere con etichette assegnate

```
accuracy = mean(labelAssegnate ==  
labelVere)
```

I PRTools mettono a disposizione una funzione generale per calcolare l'accuratezza o l'errore

Help testc

TESTC Test classifier, simple test routine

```
[E,F] = TESTC(A*W)
[E,F] = TESTC(A,W)
[E,F] = A*W*TESTC
```

INPUT

A Dataset

W Trained classifier mapping

OUTPUT

E Estimated error rate

F Number of erroneously classified
objects per class

La funzione restituisce due valori: l'errore e il numero di pattern sbagliati.

```
[error, numError] = testc(ds_banana*W);  
accuracy = 1-error
```

RIEPILOGO

Creazione dataset	<code>ds = dataset(features, labels)</code>
Plot	<code>scatterd(ds)</code>
creazione dataset sintetici	<code>gendatd, gendats, ecc</code>
Split	<code>gendat ()</code>
Accesso ai dati	
- data (features)	<code>double(ds), +ds</code>
- labels	<code>getlab(ds)</code>
- elenco features	<code>getfeat(ds)</code>
- elenco label	<code>getlablist(ds)</code>
- numerosità delle classi	<code>classsizes(ds)</code>

Creazione/ addestramento del classificatore	<code>W = fClassifier(ds,...)</code>
classificazione	<code>ds*W</code>
label assegnate	<code>labeld(ds*W);</code>
prob a posteriori	<code>+classc(ds*W);</code>
errore	<code>testc(ds,W);</code>
errore - specifico per il KNN	<code>testk</code>
matrice di confusione	<code>confmat(trueLabel, label)</code>
Plot superfici discriminanti	<code>Plotc(W)</code>

NOTA

È possibile creare un dataset non addestrato (incapsulo l'algoritmo) e addestrarlo successivamente mediante l'operatore *

Esempio

A=gendatb

W_untrained=ldc([]) % in genere, W_untrained=f([], parametri)

W=A*W_untrained

Questo è importante perchè in questo modo possiamo creare delle funzioni che accettano come input anche un classificatore non addestrato

Esercizio 1:

1. Creare un dataset artificiale (es utilizzando GENDATB, GENDATH, ...) e dividerlo in due parti (training set e test set), utilizzando la funzione GENDAT .
- esempio: banana, 400 pattern totali, il 40% nel training set
2. Costruire tramite i PRTools un classificatore lineare sul training set
3. Visualizzare i data set (training set e test set) e le superfici di decisione.
4. Classificare training set e test set
5. Calcolare accuratezza su training e test set.

Esercizio 2:

scrivere una funzione con le seguenti caratteristiche:

INPUT

- un training set A
- un test set B
- un classificatore W

la funzione addestra W su porzioni via via maggiori di A fino ad utilizzare tutto A; per ogni ciclo di addestramento produce un grafico della superficie discriminante prodotta

OUTPUT

l'errore sulla frazione di A utilizzata e su B a ciascuna iterazione

Esercizio 3:

Utilizzare la funzione precedentemente creata per mostrare l'errore di classificazione del dataset PHONEME al variare del training set

Il procedimento è identico all'esercizio 1:

```
clear

%divido il dataset
[ds_train, ds_test] = gendat(phoneme_ds1,
0.6);

% creo il classificatore
W = ldc(ds_train);
. . .
```

Matrice di confusione

La matrice di confusione permette di valutare come sono distribuiti gli errori e le decisioni corrette effettuate dal nostro classificatore.

Nella matrice di confusione le righe rappresentano le classi vere, le colonne le classi assegnate

	ω_A	ω_B	ω_C	
ω_A	20	0	0	20
ω_B	0	30	0	30
ω_C	0	0	18	18
	20	30	18	

Problema a tre classi; nessun errore di classificazione

	ω_A	ω_B	ω_C	
ω_A	15	4	1	20
ω_B	6	20	4	30
ω_C	0	8	10	18
	21	32	15	

Problema a tre classi con errori di classificazione

Vediamo come calcolare la matrice di confusione

```
% calcolo matrice di confusione
numPattern = size(ds_banana,1);
numClass = max(labelVere);

% Creo una matrice vuota
confmat = zeros(numClass + 1);

% Calcolo elementi matrice di confusione
for i=1:numPattern
    confmat(labelVere(i),labelAssegnate(i)) = ...
confmat(labelVere(i),labelAssegnate(i))+1;
end;

% Calcolo l'orlatura
confmat(end,:)=sum(confmat,1); % Sommo lungo la PRIMA
dimensione (righe)
confmat(:,end)=sum(confmat,2); % Sommo lungo la SECONDA
dimensione (colonne)
```