



## *Sommario:*

- *Matrice di confusione*
- *Cross validation*
- *Prestazioni nel KNN*



# Matrice di confusione



Il semplice calcolo dell'errore di classificazione non permette di capire il tipo di errori commesso dal nostro sistema.

In particolare non possiamo rispondere a domande come:

- Gli errori sono distribuiti equamente fra tutte le classi?
- Il sistema “confonde” solo alcune classi mentre classifica correttamente le altre?

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



# Matrice di confusione



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

Etichette assegnate dal classificatore

Classi vere

	$\omega_A$	$\omega_B$	$\omega_C$	
$\omega_A$	<b>20</b>	0	0	<b>20</b>
$\omega_B$	0	<b>30</b>	0	<b>30</b>
$\omega_C$	0	0	<b>18</b>	<b>18</b>
	<b>20</b>	<b>30</b>	<b>18</b>	

Problema di classificazione a tre classi senza errori di classificazione

Etichette assegnate dal classificatore

Classi vere

	$\omega_A$	$\omega_B$	$\omega_C$	
$\omega_A$	<b>15</b>	4	1	<b>20</b>
$\omega_B$	6	<b>20</b>	4	<b>30</b>
$\omega_C$	0	8	<b>10</b>	<b>18</b>
	<b>21</b>	<b>32</b>	<b>15</b>	

Problema di classificazione a tre classi con errori di classificazione



# Esempio costruzione dataset



I PRTools mettono a disposizione una funzione per il calcolo della matrice di confusione

```
[C,NE,LABLIST] = CONFMAT(LAB1,LAB2,METHOD,FID)
```

```
[C,NE,LABLIST] = CONFMAT(D,METHOD)
```

Dove

- LAB1,LAB2 : label da confrontare
- D : risultato della classificazione  $D = DS*W$
- C : matrice di confusione
- NE : numero di errori

Per gli altri dettagli vedi help



# Esempio costruzione dataset



## Esempio utilizzo

```
clear
ds_banana = gendatb(200);
labelVere = getlab(ds_banana);
    % Estraggo le etichette vere
W = ldc(ds_banana);
labelAssegnate = labeld(ds_banana*W);
    % Stimo le etichette
[C NE] = confmat(ds_banana*W)
```

## Output:

```
C =
    92    16
    10    82

NE =
    26
```



# Stima delle prestazioni



La stima delle prestazioni dipende dai dati utilizzati  
Pertanto dividere semplicemente i dati casualmente fra training e test set non garantisce che i risultati siano statisticamente significativi.

La ripetizione della valutazione su differenti divisioni casuali e il calcolo delle prestazioni in termini di media e dev.standard delle singole valutazioni consente di avere una stima più affidabile.

```
for rep=1:nRep
    ...
    err(rep) = ...
end

errAvg = mean(err)
errStd = std(err)
```



Tuttavia anche ripetizione delle valutazioni su differenti divisioni casuali potrebbe impedire di utilizzare in fase di valutazione (o di addestramento) i dati più complessi da classificare.

Soluzione: **Cross Validation**



# Cross validation



Nella Cross Validation tutti i dati a disposizione vengono utilizzati, a gruppi di dimensione fissa, alternativamente come test e come training set.

Pertanto ogni pattern viene sia classificato (almeno una volta) sia utilizzato per l'addestramento.





# Cross validation



- Considero un dataset.



# Cross validation



**F1**

**F2**

**F3**

**F4**

**F5**

- Considero un dataset.
- Divido il dataset casualmente in N fold (es in 5 parti)



# Cross validation



**F1**

**F2**

**F3**

**F4**

**F5**

- Considero un dataset.
- Divido il dataset casualmente in N fold (es in 5 parti)
- Prendo a turno un fold come test e i rimanenti come train set

Es:

**Iter1:**  $tst = F1$ ;  $trn = F2 + F3 + F4 + F5$

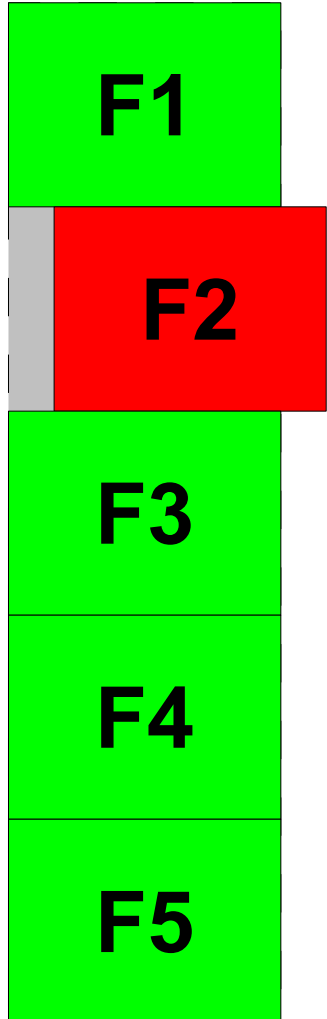
**Iter2:**  $tst = F2$ ;  $trn = F1 + F3 + F4 + F5$

...

**Iter5:**  $tst = F5$ ;  $trn = F1 + F2 + F3 + F4$



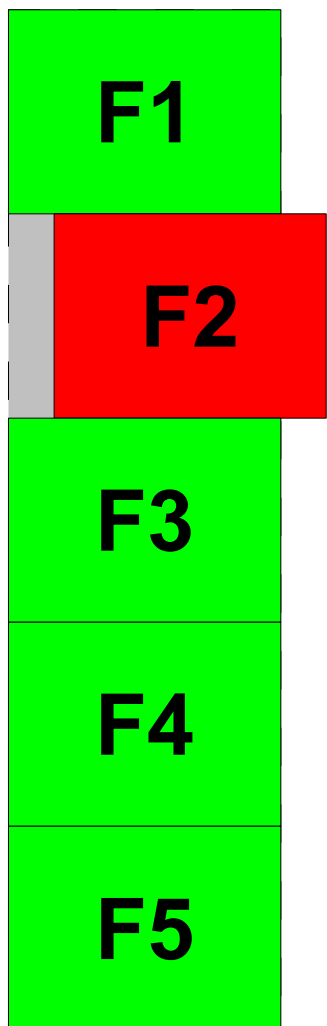
# Cross validation



- Considero un dataset.
- Divido il dataset casualmente in N fold (es in 5 parti)
- Prendo a turno un fold come test e i rimanenti come train set
  - Addestro sul train set (es:  $\text{trn} = F1 + F3 + F4 + F5$ )
  - Classifico il test (es  $\text{tst} = F2$ )



# Cross validation



- Considero un dataset.
- Divido il dataset casualmente in N fold (es in 5 parti)
- Prendo a turno un fold come test e i rimanenti come train set
  - Addestro sul train set (es:  $\text{trn} = F1 + F3 + F4 + F5$ )
  - Classifico il test (es  $\text{tst} = F2$ )
- Valuto le prestazioni come media fra le prestazioni ottenute su ciascuno degli N fold



**F1**

**F2**

**F3**

**F4**

**F5**

## Alcune considerazioni:

Al termine delle iterazioni tutti pattern del dataset (a gruppi) saranno stati classificati.

Le prestazioni ottenute dipendono comunque dalla particolare divisione in fold. Pertanto può essere utile ripetere la cross validation più volte in modo da rendersi indipendenti dalla particolare divisione.



# Cross validation



I PRTools permettono di eseguire la cross validation con e senza ripetizione con un qualsiasi classificatore PRTools.

Il classificatore non deve essere addestrato ma può essere inizializzando tramite i parametri specifici

VEDI help crossval

```
% creo un classificatore non addestrato  
W = ldc([], %parametri% )
```



# Cross validation - Esempio



Generate un dataset e valutare le prestazioni di un classificatori (es LDC,QDC,...) con 5 fold, con e senza ripetizioni

```
ds_banana = gendatb(200) ;  
w=qdc([], 0.6) ;  
ERR = crossval(ds_banana,w,5,1)  
%valuto con ripetizioni  
[ERR,STDS] = crossval (ds_banana,w,5,20)
```

Output:

```
ERR =  
    0.1996
```

```
ERR =  
    0.2085
```

```
STDS =  
    0.0040
```





# Cross validation - Esercizio



Generate un dataset artificiale e valutare le prestazioni di un classificatore (es LDC, QDC, ...) nei seguenti casi:

- usando divisione casuale in train (70%) e test (30%)
- usando cross-val con diversi numeri di fold: 2, 5, 10
- Ripetere 10 volte ciascuno dei 4 casi e riportare media e varianza dell'errore

N.B. Si raccomanda l'utilizzo degli script.



# Cross validation - Esercizio



Soluzione (1p):

```
clear
dsFull = gendatb(100,1.5);
nRep = 10;
nFold = [2 5 10]';

% Stima con divisione in train e test
for iRep=1:nRep
    [dsTrn dsTst] = gendat(dsFull, 0.70);
    W = qdc(dsTrn);
    errorSplit(iRep) = testc(dsTst*W);
end

disp('Split [mean std]:')
disp([mean(errorSplit) std(errorSplit)])
```



# Cross validation - Esercizio



Soluzione (2p):

```
% Stima con cross validation
W = qdc;
for k=1:length(nFold),
    [errorXVal(k,1) errorXValSTD(k,1)] = ...
        crossval(W, dsFull, nFold(k),nRep);
end

disp('CrossVal [nFold mean std]:')
disp([nFold errorXVal errorXValSTD])
```



# Stima prestazioni K-NN



I PRTools mettono a disposizione una funzione specifica (TESTK) per la stima dell'errore del classificatore K-NN.

```
E = TESTK(A,K,T)
```

## INPUT

A	Training dataset
K	Number of nearest neighbors (default 1)
T	Test dataset (default [], i.e. find leave-one-out estimate on A)

## OUTPUT

E	Estimated error of the K-NN rule
---	----------------------------------

Sfruttando le peculiarità del K-NN la funzione TESTK permette di eseguire in modo efficiente un particolare caso di Cross Validation denominato LEAVE-ONE-OUT, in cui ciascun fold contiene un solo pattern (N-Fold CrossVal. con  $N = \text{num.patterns}$ ).



Sfruttando le peculiarità del K-NN la funzione TESTK permette di eseguire in modo efficiente un particolare caso della Cross Validation denominato LEAVE-ONE-OUT

LEAVE-ONE-OUT == N-Fold Cross Validation  
con  $N = \text{num.pattern}$

(Ciascun fold contiene un solo pattern)

NOTA:

La funzione knnc è in grado di utilizzare il leave-one-out per la stima del valore ottimale del parametro  $k$  sul training set.



# Stima prestazioni KNN



Confrontiamo i tempi di calcolo della funzione standard con quella specifica del KNN

```
clear
ds = gendatb(200);
W = knnc([],5);
tic; err = crossval(ds, W,100), toc
tic; err = testk(ds,5), toc
```

Output:

```
err = 0.0348
Elapsed time is 0.960034 seconds.
```

```
err = 0.0350
Elapsed time is 0.014741 seconds.
```