

# Valutazione dei risultati della classificazione

Gianluca Amato

Corso di Laurea Specialistica in Economia Informatica  
Università “G. D'Annunzio” di Chieti-Pescara  
ultimo aggiornamento: 12/05/08

Tasso di errore vero e campionario

# Tasso di errore vero

- Bisogna stimare l'accuratezza di un classificatore:
  - ci serve una misura della prestazione del classificatore
  - per i problemi di classificazione, una misura naturale è il tasso di errore.
- Supponiamo che
  - $I$  sia l'insieme di tutte le istanze possibili
  - $c(I)$  è la classe “vera” di  $I$
  - $h(I)$  è la classe “predetta” di  $I$
- Si definisce allora il **tasso di errore vero** (**true error rate**) come

$$error_S(h) = \frac{1}{n} \sum_{i \in I} \delta(h(i), c(i))$$

errore vero

dove  $\delta(h(i), c(i))$  è 0 se  $h(i)=c(i)$ , 1 altrimenti.

# Tasso di errore su un campione

- Nella maggior parte dei casi  $I$  non è utilizzabile perché ignoto o troppo grande.
- Si considera allora un campione  $S \subseteq I$  e si calcola il **tasso di errore campionario su  $S$** .

$$error_S(h) = \frac{1}{n} \sum_{i \in S} \delta(h(i), c(i))$$

errore campionario

- Si stima  $error_I(h)$  sulla base di  $error_S(h)$ 
  - sotto opportune ipotesi,  $error_S(h)$  è “vicino” a  $error_I(h)$
- Invece del tasso di errore, si può definire il **tasso di successo o accuratezza**.

# Errore di sostituzione

- Il modo più immediato per scegliere  $S$  è usare l'insieme di addestramento.
  - si parla di **errore di sostituzione** (**resubstitution error**)
- La stima è troppo ottimistica!!
  - occorre utilizzare due insiemi distinti, uno per l'addestramento e l'altro per il test.
- È importante che l'insieme di test non sia utilizzato in alcun modo nella fase di costruzione del modello
  - ad esempio, nella fase di “reduced error pruning” non va usato l'insieme di test, ma un terzo insieme di dati, indipendente sia da quello di addestramento che da quello di controllo.
- Una volta compiuta la stima, si può rimettere l'insieme di test dentro quello di addestramento, e ricalcolare il modello.

# Stima puntuale e per intervalli

- Quando l'insieme  $S$  è scelto in maniera indipendente da quello di addestramento, allora  $\text{error}_S(h)$  è uno stimatore **non distorto** e **consistente** di  $\text{error}_I(h)$ .
- Possiamo essere interessati però a determinare degli **intervalli di confidenza**.
- Esempio 1:
  - $\text{error}_S(h)=25\%$ ,  $|S|=1000$
  - con confidenza dell'80%,  $\text{error}_I(h) \in [0.233, 0.268]$
- Esempio 2:
  - $\text{error}_S(h)=25\%$  ma  $|S|=100$
  - con confidenza dell'80%,  $\text{error}_I(h) \in [0.203, 0.313]$

# Processi di Bernoulli e binomiale

- Supponiamo di estrarre a caso una delle  $I$  istanze
  - consideriamo la variabile aleatoria  $E: I \rightarrow \{0,1\}$  data da  $E(i) = \delta(h(i), c(i))$
- La variabile  $E$  ha una distribuzione di Bernoulli con parametro  $p = \text{error}_I(h)$ 
  - media =  $p$
  - varianza =  $p(1-p)$
- $n * \text{error}_S(h)$  è semplicemente la somma di  $n$  variabili di Bernoulli indipendenti, dunque ha una distribuzione binomiale
  - media =  $np$
  - varianza =  $np(1-p)$

# Lo stimatore $error_S(h)$

- Quindi, su  $error_S(h)$  possiamo dire che ha
  - media:  $error_I(h)$
  - varianza:  $1/n * error_I(h) (1-error_I(h))$
- Questo dimostra che  $error_S(h)$  è uno **stimatore non distorto** di  $error_I(h)$
- Vogliamo calcolare un intervallo di confidenza per  $error_S(h)$ 
  - ovvero, dato un livello di confidenza  $c$ , vogliamo trovare un valore  $z$  tale che

$$Pr\{|error_S(h) - error_I(h)| \leq z\} = c$$

ovvero

$$Pr\{error_S(h) - z \leq error_I(h) \leq error_S(h) + z\} = c$$



# Intervalli di confidenza per variabili normali (1)

- Il calcolo di un intervallo di confidenza per una distribuzione binomiale è faticoso
  - tuttavia, per valori di  $n$  elevati, una distribuzione binomiale di media  $p$  e varianza  $v$  è approssimabile con una distribuzione gaussiana con stessa media e varianza.
- Se  $X$  è una distribuzione gaussiana di media 0 e varianza 1 (si chiama distribuzione normale), esistono tabelle (e algoritmi) che danno il valore di
  - $\Pr \{ X \geq z \}$
- Siccome  $X$  è simmetrica rispetto all'asse  $y$ , allora
  - $\Pr \{ |X| \leq z \} = \Pr \{ -z \leq X \leq z \} = 1 - 2 * P \{ X \geq z \}$
- Possiamo allora calcolare gli intervalli di confidenza per  $X$

# Intervalli di confidenza per variabili normali (2)

- Consideriamo la seguente tabella, dove  $X$  è una variabile normale:

$P\{X \geq z\}$	$z$
0.01%	3.72
0.50%	2.58
1.00%	2.33
5.00%	1.65
10.00%	1.28
20.00%	0.84
40.00%	0.25

- L'intervallo di confidenza per  $X$  con confidenza del 90% è  $[-1.65, 1.65]$ 
  - in quanto  $\Pr \{-1.65 \leq X \leq 1.65\} = 0.9$

# Intervalli di confidenza per $error_I(h)$

- Se  $p=error_I(h)$  e se supponiamo di approssimare la distribuzione binomiale con una gaussiana,

$$\frac{error_s(h) - p}{\sqrt{p(1-p)/n}}$$

è una variabile gaussiana di media 0 e varianza 1

- Dato un valore di confidenza  $c$ , sappiamo trovare  $z$  tale che

$$Pr \left\{ -z \leq \frac{error_s(h) - p}{\sqrt{p(1-p)/n}} \leq z \right\} = c$$

- Risolvendo su  $p$  otteniamo

$$p = \left( error_s(h) + \frac{z^2}{2n} \pm z \sqrt{\frac{f}{n} - \frac{f^2}{n} + \frac{z^2}{4n}} \right) / \left( 1 + \frac{z^2}{n} \right)$$

# Esempi di intervalli di confidenza

- Invece di un semplice esempio statico, questa è una tabella in OpenOffice Calc per determinare gli intervalli di confidenza

Dati		Intervallo risultante [p1,p2]	
errorS(h)	25,00%	p1	0,203
n	100	p2	0,313
c	0,8		
z	1,28		

- Attenzione che per valori di  $n$  molto piccoli, l'approssimazione gaussiana non è più indicata.
  - diciamo che per  $n > 100$  non c'è problema

# Metodi di valutazione

# Metodo holdout

- **Holdout**: metodo con cui si divide l'insieme di dati in una parte usata per l'addestramento e una per il testing.
  - tipicamente  $1/3$  è usato per il test e  $2/3$  per l'addestramento.
  - stima pessimistica in quanto per l'addestramento si usa un sotto-insieme del campione.
- **Problema**: il campione potrebbe non essere rappresentativo
  - ad esempio, alcune classi poco numerose potrebbero mancare nell'insieme di addestramento.
  - le versioni avanzate di holdout usano la tecnica della **stratificazione**.
    - le proporzioni tra le varie classi nell'insieme di dati originario devono essere le stesse che si riscontrano negli insiemi di test e di addestramento.
  - il problema comunque rimane se il numero di istanze è esiguo.

# Validazione incrociata

- Si divide l'insieme di dati in  $k$  parti
  - ripeto, per tutti i valori di  $i$  da 1 a  $k$ , i seguenti passi:
    - addestro il sistema con tutti i dati tranne quelli della partizione  $i$
    - uso la partizione  $i$  per calcolare il tasso di errore
  - calcolo il tasso di errore finale dividendo il numero di errori totale nei  $k$  passi per il numero di istanze totali.
  - si parla di **k-fold cross validation**.
- Che valore scegliere per  $k$ ?
  - da numerosi esperimenti e da alcuni risultati teorici si è visto che 10 è un buon valore.
- I singoli fold possono essere scelti stratificati.

# Leave one out cross validation (1)

- Se si sceglie  $k=N$  (numero totale di istanze), si ha la **leave one out cross validation**.
- Vantaggi:
  - fa il massimo uso dei dati a disposizione
  - non ci sono campionamenti casuali
- Svantaggi:
  - computazionalmente oneroso (tranne che per il metodo k-NN) in quanto è necessario addestrare il sistema ben  $N$  volte.



# Leave one out cross validation (2)

- Caso estremo:
  - insieme di istanze  $I$ , composto da due classi della stessa numerosità.
  - consideriamo un insieme di dati  $S$  ricavato da  $I$ , con lo stesso numero di elementi per le due classi, e supponiamo di applicare il metodo zeroR.
  - abbiamo che
    - ad ogni passo del LOO-CV, la classe opposta alla istanza di test è maggioritaria
    - la predizione sarà sempre scorretta, dando un errore stimato del 100%
  - ...ma il tasso di errore vero è del 50% (**stima molto pessimistica**)

# Bootstrap

- I metodo di validazione incrociata e di holdout usano un campionamento senza rimpiazzo
  - una volta che una istanza è stata scelta per una determinata partizione, non può essere scelta di nuovo
- I metodi di **bootstrap** usano invece un campionamento con rimpiazzo
  - da un insieme di dati di  $N$  istanze, ne vengono scelte  $N$  con rimpiazzo
    - le istanze scelte fanno parte dell'insieme di addestramento
    - quelle che non sono mai state scelte fanno parte dell'insieme di testing.
  - si ripete il procedimento con campioni differenti
  - si tenta di recuperare la relazione tra l'insieme di istanze di addestramento  $S$  e l'insieme di tutte le istanze  $I$ , sfruttando la relazione tra  $S$  e i suoi sotto-campioni

# Bootstrap (2)

- In particolare esaminiamo il metodo del **0.632 bootstrap**.
  - ogni istanza ha probabilità  $1 - 1/N$  di non essere scelta
  - pertanto la sua probabilità di far parte dell'insieme di test è

$$\left(1 - \frac{1}{N}\right)^N \simeq e^{-1} \simeq 0.368$$

dal limite notevole  $\lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n = e^x$

- L'insieme di addestramento conterrà quindi circa il 63.2% delle istanze.

# Bootstrap (3)

- La stima del tasso di errore ottenuto dal bootstrap è pessimistica
  - l'insieme di addestramento contiene solo il 63.2% delle istanze (contro il 90% del 10-fold CV e il quasi 100 della LOO-CV)
- Spesso, la si bilancia con l'errore di sostituzione, ottenendo

$$err = 0.632 \cdot err_{\text{istanze di test}} + 0.368 \cdot err_{\text{istanze di addestramento}}$$

- Il procedimento si ripete varie volte e si mediano i risultati.

# Bootstrap (4)

- Si è rivelato **molto efficace per piccoli insiemi di dati** ma ha dei casi limite:
  - I composto da due classi di uguale numerosità e con attributo classe indipendente dagli altri attributi
  - un modello che
    - in fase di addestramento memorizza tutte le istanze
    - in fase di utilizzazione, se ha informazioni sulla istanza in input, risponde con il valore memorizzato, altrimenti risponde a caso
  - se  $S$  è abbastanza piccolo rispetto ad  $I$ , allora il tasso di errore vero è circa il 50%
  - il tasso di errore è lo 0% sull'insieme di addestramento, il 50% su un insieme di test indipendente
  - il metodo del bootstrap stima per questo classificatore un tasso di errore del  $0.632 * 50\% = 31.6$  (**eccessivamente ottimistico**).

## Altre misurazioni di errore

# Matrice di confusione

- Tutti gli errori commessi dal classificatore possono essere visualizzati in una **matrice di confusione**:

<i>classi vere\predette</i>	<i>Classe 1</i>	<i>Classe 2</i>	<i>Classe 3</i>	<i>...</i>	<i>Classe k</i>
<i>Classe 1</i>	23	3	4		2
<i>Classe 2</i>	..	17	..		..
<i>Classe 3</i>	..	..	15		..
<i>....</i>					
<i>Classe k</i>	..	..	..		93

- L'errore campionario si ottiene dalla somma dei valori di tutte le celle, esclusa la diagonale.

# Caso con due classi

- Se il nostro attributo classe assume solo due valori, che indichiamo con **true** e **false**, tutte le istanze dell'insieme di test si possono distinguere in quattro gruppi:
  - TP (**true positive**): istanze di classe true classificate come true;
  - FP (**false positive**): istanze di classe false classificate incorrettamente come true;
  - TN (**true negative**): istanze di classe false classificate come false;
  - FN (**false negative**): istanze di classe true classificate incorrettamente come false;
- Ogni cella della matrice di confusione corrisponde al numero di uno di questi casi.
- L'errore campionario è  $FP+FN$ .



# La statistica K (1)

- Un'alternativa al tasso di errore, soprattutto nel caso di classi con frequenza sbilanciata, è la **statistica K**.
- Questa statistica tiene conto della probabilità che il nostro classificatore indovini la classe giusta per pura coincidenza:

$$K = \frac{P(A) - P(E)}{1 - P(E)}$$

dove

- P(A) è la probabilità che il classificatore indovini (ovvero l'accuratezza);
- P(E) è la probabilità che il classificatore indovini per pura coincidenza
  - vedremo nelle prossime slide come calcolarla

# La statistica K (2)

- La statistica **K** assume i valori tra  $-\infty$  ed 1
  - il valore 1 si ottiene solo quando  $P(A)=1$ ;
  - il valore 0 corrisponde ad un classificatore che indovina solo grazie al caso;
  - valore minori di 0 corrispondono a classificatori che si impegnano per sbagliare le previsioni.
- Come **calcolare P(E)**?
  - probabilità delle classi  $C_1.. C_k$  nell'insieme di test:  $p_1.. p_k$
  - probabilità delle classi  $C_1.. C_k$  nell'output del nostro classificatore:  $q_1...q_k$ .
  - allora 
$$P(E) = \sum_{i=1}^k p_i \cdot q_i$$

# La statistica K (3)

- Esempio 1
  - la classe A rappresenta il 90% delle istanze e la classe B il 10%
  - il classificatore predice tutte le istanze come classe A
  - tasso di errore: 10%
  - l'accuratezza: 90%
  - $P(E) = 0.9 * 1 + 0.1 * 0 = 0.9 = 90\%$
  - $K = 0$  (dunque il classificatore indovina solo per caso)

# La statistica K (4)

- Esempio 2

- la classe A rappresenta il 90% delle istanze e la classe B il 10%
- il classificatore ha la seguente **matrice di confusione**:

	A	B	
A	0.75	0.05	classe vera
B	0.15	0.05	

classe predetta

- tasso di errore: 20% (dunque peggiore di quello di prima)
- accuratezza: 80%
- $P(E) = 0.9 * 0.8 + 0.1 * 0.2 = 0.74 = 74\%$
- $K = (0.8 - 0.74) / (1 - 0.74) = 0.23 = 23\%$

# Valutazione di previsioni probabilistiche

# Accuratezza di predizioni probabilistiche

- Finora abbiamo usato come misura di accuratezza il tasso di errore (o il tasso di successo)
- La maggior parte dei classificatori possono essere modificati per produrre distribuzioni di probabilità
  - vorremmo tener conto di questa distribuzione per valutare l'errore
  - se la predizione è che al 90% la classe di una istanza  $x$  è true, mentre in realtà la classe di  $x$  è false, siamo di fronte a un errore più grave che se la probabilità prevista fosse stata solo del 60%.
- Decidere se tenere conto o no delle probabilità nella stima della bontà di un classificatore dipende dall'utilizzo che se ne fa
  - ad esempio, se la stima verrà vagliata da un esperto

# Funzioni di perdita

- Siano  $C_1, \dots, C_k$  le possibili classi: si chiama **funzione di perdita** una funzione a valori reali con due parametri:
  - una distribuzione di probabilità su  $C_1 \dots C_k$
  - un intero da 1 a  $k$  che indica la vera classe dell'istanza
- data una istanza  $i \in I$ 
  - $h(i)$  è la **distribuzione di probabilità** calcolata dal classificatore (e non più la classe predetta come visto fin'ora)
  - $c(i)$  è il vero valore della istanza  $i$
- data un funzione di perdita  $\delta$ , l'**errore vero rispetto a  $\delta$**  è

$$error_I(h) = \frac{1}{I} \sum_{i \in I} \frac{\delta(h(i), c(i))}{k}$$

dove la media è calcolata sulla distribuzione di probabilità su  $I$ .

# Errore vero ed errore campionario

- L'errore vero non è calcolabile, e si approssima quindi con l'errore campionario.
- Dato  $S \subseteq I$ , definiamo l'errore sul campione

$$error_S(h) = \frac{1}{n} \sum_{i \in S} \frac{\delta(h(i), c(i))}{k}$$

- Vari errori a seconda della funzione di perdita
  - errore assoluto medio
  - errore quadratico medio
  - etc...



# Funzione di perdita quadratica (1)

- Una delle funzioni di perdita più utilizzate è la **funzione di perdita quadratica**

$$\delta(p, c) = \sum_{j=1}^k (p_j - a_j)^2$$

dove  $a_j = 1$  se  $j=c$ , altrimenti  $a_j = 0$ , ovvero

$$\delta(p, c) = \sum_{j \neq c} p_j^2 + (1 - p_c)^2$$

- Una funzione simile è la funzione di perdita assoluta:

$$\delta(p, c) = \sum_{j=1}^k |p_j - a_j|$$

# Funzione di perdita quadratica (2)

- Giustificazione

- supponiamo che la variabile classe sia casuale e indipendente dagli altri attributi
- minimizzare la perdita quadratica vuol dire utilizzare un classificatore che produce come output  $p$  la vera distribuzione di probabilità  $p^*$  delle classi

- Infatti

$$\begin{aligned} error_I(h) &= E\left[\sum_j (p_j - a_j)^2\right] = \sum_j (E[p_j^2] - 2E[p_j a_j] + E[a_j^2]) = \\ &= \sum_j (p_j^2 - 2p_j p_j^* + p_j^*) = \sum_j ((p_j - p_j^*)^2 + p_j^*(1 - p_j^*)) \end{aligned}$$

# Funzione di perdita di informazione

- Nella teoria dei codici, se  $c$  è un simbolo che occorre con probabilità  $p_c$ , il numero di bit necessari alla sua rappresentazione è  $\log_2 p_c$ 
  - supponendo di usare il miglior codice possibile

- Definisco sulla base di questa osservazione la funzione di perdita:

$$\delta(p, c) = -\log_2 p_c = -\sum_{j=1}^k a_j \log_2 p_j$$

- Nel caso di attributo classe casuale e indipendente

$$\text{error}_I(h) = E\left[-\sum_j a_j \log_2 p_j\right] = -\sum_j p_j^* \log_2 p_j$$

e studiando questa funzione si vede che è minimizzata per  $p=p^*$

# Quale funzione di perdita scegliere?

- Al solito, dipende dalla situazione
- La perdita quadratica
  - prende in considerazione tutte le classi
  - non può mai essere superiore a 2
- La perdita di informazione
  - considera solo la probabilità della classe corretta
  - può essere **infinita** quando la probabilità di una classe è 0
  - è legata al problema della rappresentazione delle informazioni e al “*minimum description length principle*”
- **MDL principle**: il metodo di classificazione migliore è quello che minimizza il numero di bit necessari per codificare il modello e i dati

# Errori relativi

- Oltre agli errori visti fino ad ora, si considerano spesso gli **errori relativi**.
- Si ottengono confrontando l'errore del modello ottenuto con l'errore del **classificatore elementare**.
  - il **classificatore elementare** è quello che attribuisce alla classe  $C_k$  probabilità pari alla sua frequenza relativa nell'insieme di istanze di addestramento
- **errore relativo quadratico**: ottenuto dividendo l'errore quadratico medio per l'errore quadratico medio del classificatore elementare
  - analogamente per gli altri tipi di errore.

# Il costo degli errori

# Costo degli errori

- Gli errori che si commettono non sempre hanno la stessa gravità:
  - diagnosi precoce di un guasto
  - predizione macchie di petrolio
- In particolare, negli esempi qui sopra, la maggior parte delle volte non c'è nessuna situazione “degenere”:
  - il classificatore che sceglie sempre la classe false (nessun guasto, nessuna macchia) ha un accuratezza altissima
- Si vorrebbero dunque pesare, in maniera diversa, celle diverse della matrice di confusione:
  - pesare nella fase di valutazione (facile)
  - pesare nella fase di apprendimento

# Apprendimento sensibile ai costi

- La maggior parte degli schemi di apprendimento non sono sensibili ai costi
  - si pensi agli algoritmi basati sugli alberi di decisione
- Tecniche standard per ottenere metodi sensibili ai costi in una situazione a **due classi**:
  - Ricampionare i dati in accordo ai costi
    - se il costo di non accorgersi di una macchia di petrolio è più alto di quello di dare un falso allarme, ricampiono i miei dati in modo che le istanze di addestramento con vere macchie di petrolio siano in numero maggiore di quelle senza macchie
  - Pesare le istanze in accordo ai costi
    - abbiamo visto che gli alberi di decisioni pesano le istanze quando devono gestire attributi mancanti
    - la stessa cosa si può fare per assegnare pesi diverse alle istanze, a seconda della classe della stessa.



# Stimare i costi

- In realtà, raramente si ha una idea precisa di quanto siano i costi dei vari errori
  - dipende anche da fattori difficili da ponderare, come il costo di acquisire dati di addestramento
- Di solito, le decisioni vengono prese confrontando tra loro differenti scenari:
  - si vuole valutare un spedizione in massa di una offerta promozionale a un milione di abitazioni
    - stando ai dati precedenti, la proporzione che risponde positivamente all'offerta è lo 0.1% (1000 abitazioni)
    - un tool di data mining identifica un sottoinsieme di 100.000 abitazioni con tasso di risposta dello 0.4% (400 abitazioni)
    - che fare? probabilmente si vogliono valutare altri scenari: lo stesso tool con altri parametri identifica 400.000 abitazioni con tasso di risposta dello 0.8%

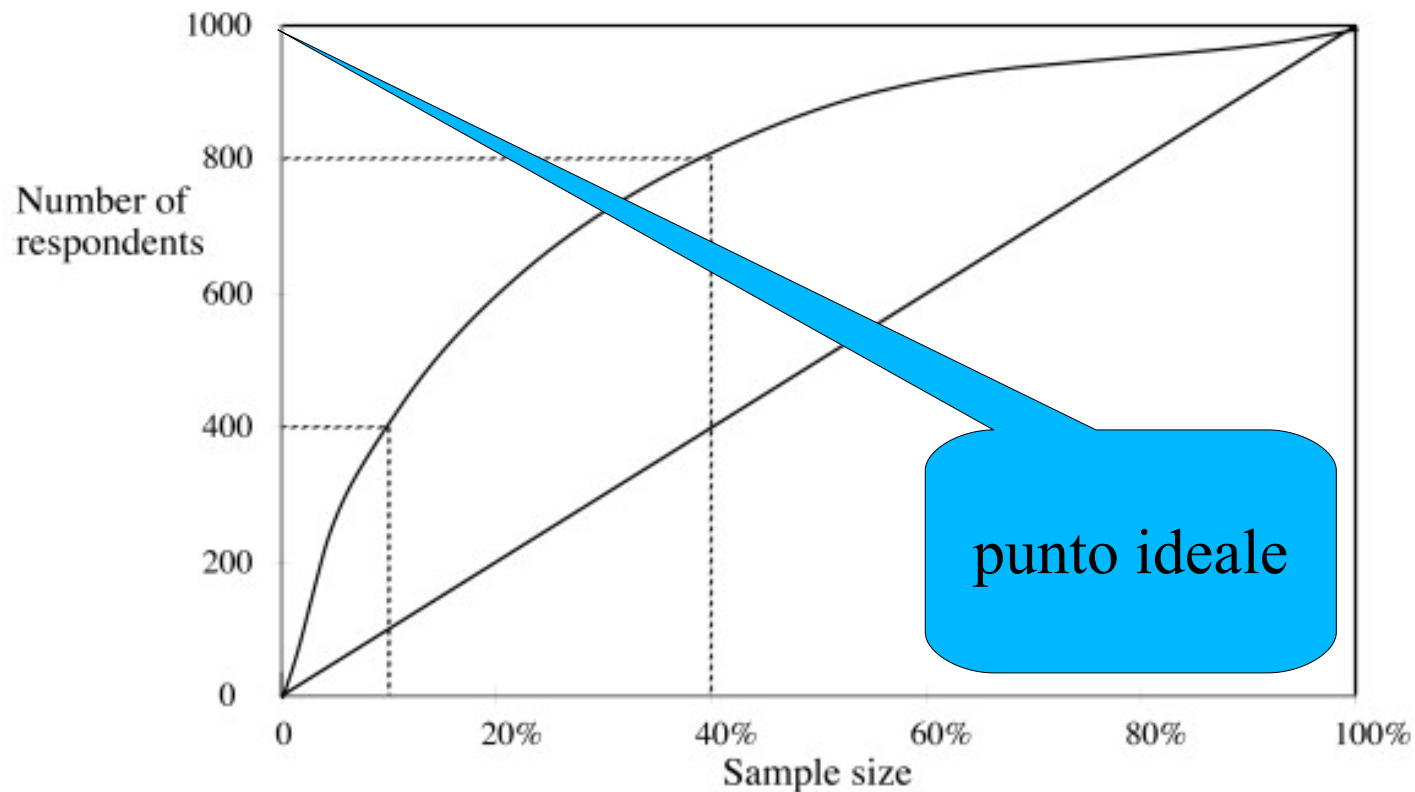
# Lift chart (1)

- Si possono visualizzare vari scenari, in una situazione a due classi, con i **lift chart**
  - nella terminologia del marketing, l'aumento del tasso di risposta visto nell'esempio precedente (dallo 0.1% allo 0.4%) è chiamato “**lift factor**”
- necessità di un algoritmo che produce una distribuzione di probabilità
  - Le istanze di test sono ordinate in ordine decrescente secondo la probabilità della classe true.

<i>Posizione</i>	<i>Probabilità</i>	<i>Classe effettiva</i>
1	0.95	yes
2	0.93	yes
3	0.93	no
4	0.88	yes
...	....	...

# Lift chart (2)

- Per ogni  $i$  tra 1 ed  $N$  (numero istanze)
  - si considera il campione costituito dalle prime  $i$  istanze
  - si traccia un punto su un piano cartesiano dove
    - l'asse x è la frazione  $i/N$  in percentuale (la dimensione del campione)
    - l'asse y è il numero di veri positivi in quel campione

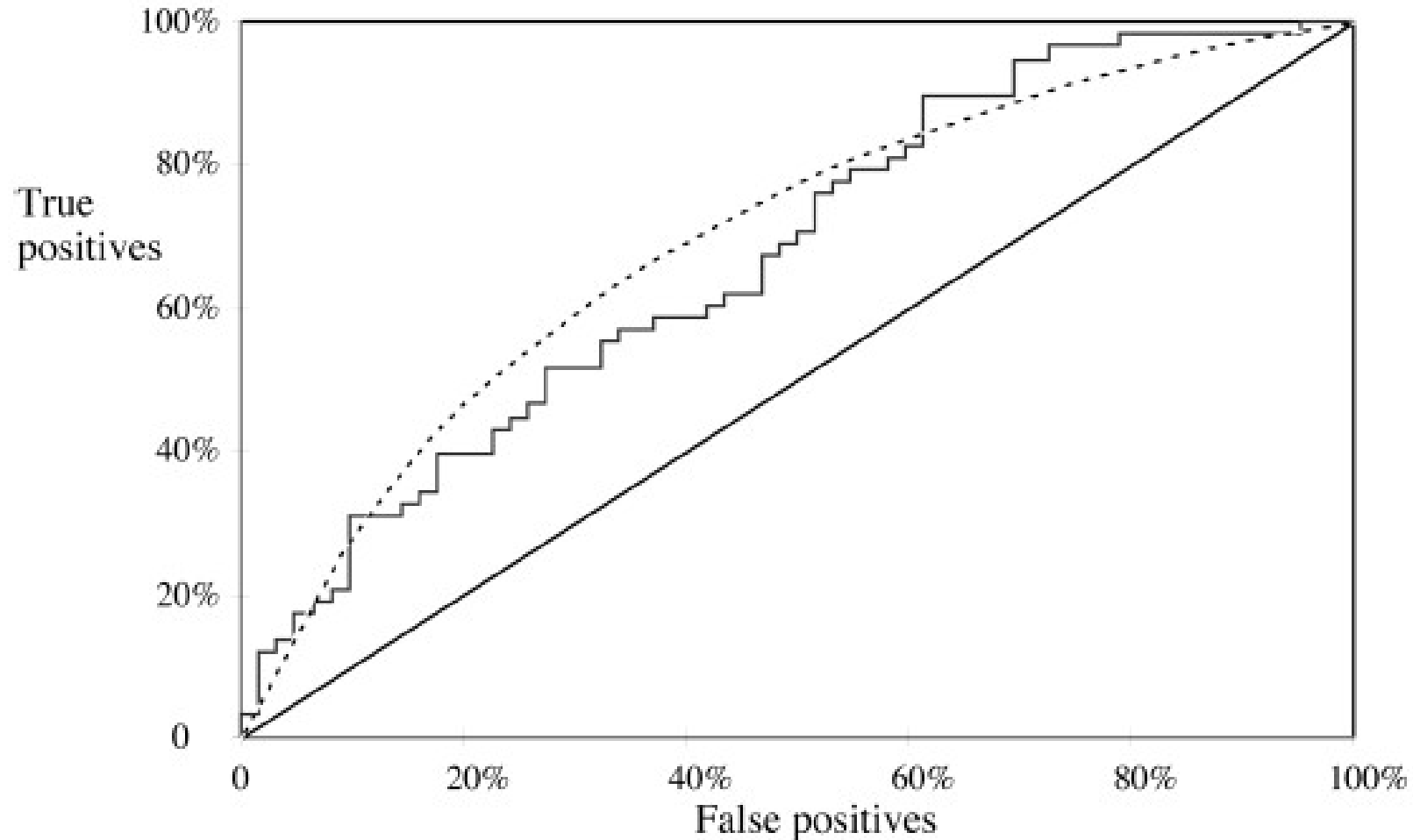


# Curve ROC (1)

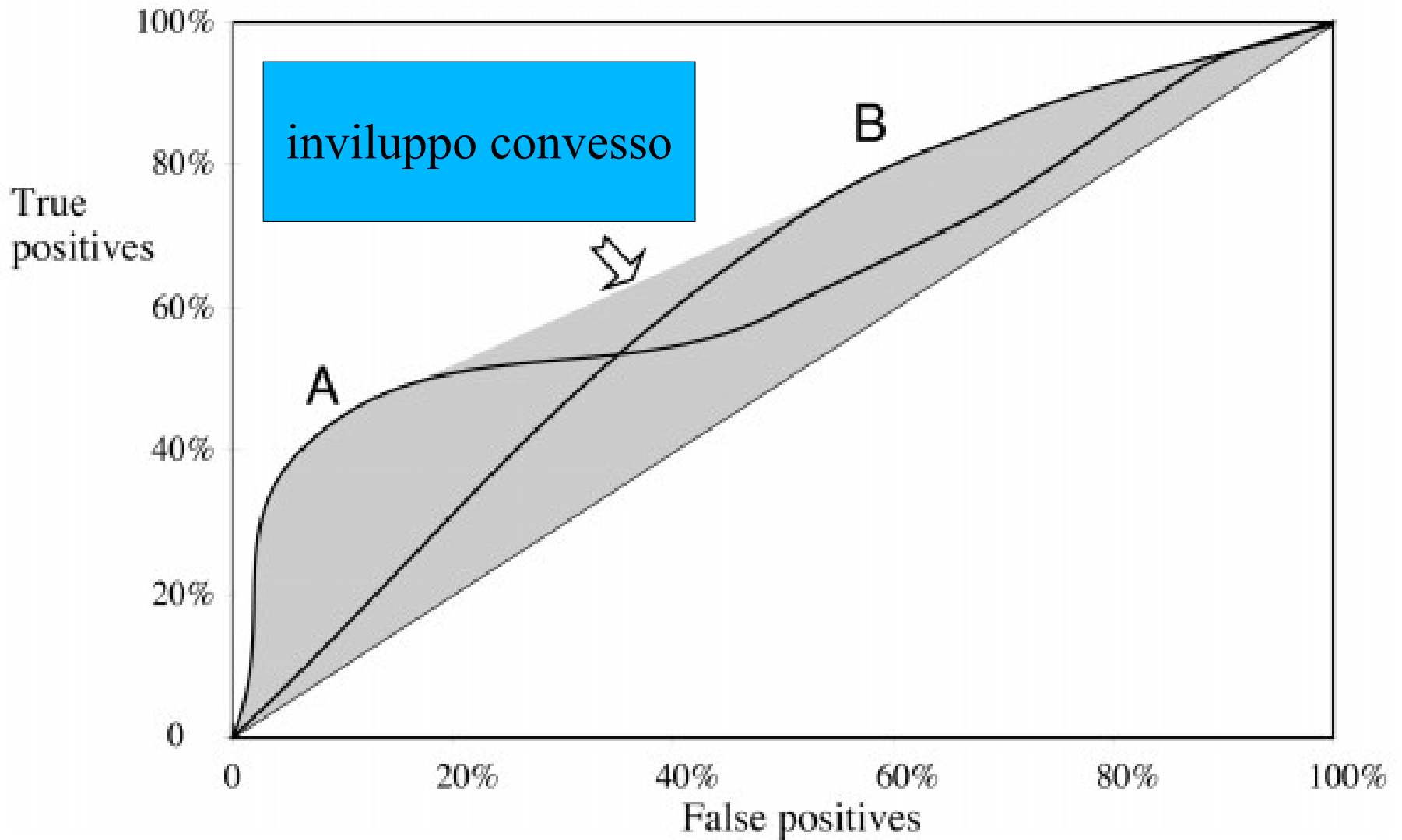
- Le **curve ROC** sono simili ai lift chart
  - ROC sta per “**receiver operating characteristics**”
  - usato nel riconoscimento di segnali per mostrare il trade off tra numero di errori riconosciuti e numero di falsi allarmi in un canale di trasmissione rumoroso
- Differenze con i lift chart
  - l'asse x è la percentuale di falsi positivi (rispetto al numero totale di istanze con classe negativa)
  - l'asse y è la percentuale di veri positivi (rispetto al numero totale di istanze con classe positiva)
- Se il numero di veri positivi è molto basso (ad esempio nel caso di marketing diretto), non c'è molta differenza tra curve ROC e lift chart.

# Curve ROC (2)

- Esempio di curva ROC per lo stesso insieme di dati di prima



# Curve ROC per due classificatori



# L'involuppo convesso

- Dati due classificatori, possiamo raggiungere qualunque punto nell'involuppo convesso delle due curve ROC
- Siano
  - $t_1, f_1$  i valori di true positive e false positive scelti per il 1° classificatore
  - $t_2, f_2$  i valori di true positive e false positive scelti per il 1° classificatore
  - fissato un  $q \in [0,1]$  se il primo schema è usato per classificare, in maniera casuale,  $100\% * q$  dei casi e il secondo per i restanti, il nuovo schema combinato avrà
    - true positive =  $q * t_2 + (1-q) * t_1$
    - false positive =  $q * f_2 + (1-q) * f_1$
  - al variare di  $q$ , percorre tutta la retta da  $(f_1, t_1)$  a  $(f_2, t_2)$

# Valutazione di previsione numeriche



# Valutazione di predizioni numeriche

- Stesse strategie adottare per la classificazione
  - insiemi di test, cross validation, etc..
- Cambia la funzione di errore
  - se  $p(i)$  è il valore predetto per la istanza  $I$  e  $a(i)$  il valore effettivo
  - sia  $S \subseteq I$  un insieme di istanze di cardinalità  $n$
  - la funzione di errore più usata è l'**errore quadratico medio**

$$eqm = \frac{\sum_{i \in S} (p(i) - a(i))^2}{n}$$

- semplice da manipolare matematicamente
- sotto opportune ipotesi, corrisponde a massimizzare la verosimiglianza dei dati

# Altre funzioni di errore

- La **radice dell'errore quadratico medio** è espressa nella stesse unità di misura dei dati

$$reqm = \sqrt{\frac{\sum_{i \in S} (p(i) - a(i))^2}{n}}$$

- l'**errore assoluto medio** è meno sensibile agli outlier

$$ram = \frac{\sum_{i \in S} |p(i) - a(i)|}{n}$$

- talvolta in queste formule si sostituisce l'errore assoluto  $p(i) - a(i)$  con l'**errore relativo** alla previsione  $(p(i) - a(i))/p(i)$

# Misure relative alla media

- Talvolta vogliamo tener conto di quanto il nostro classificatore è migliore di quello banale che predice sempre la media dei dati
- Definiamo allora l'**errore quadratico relativo** come

$$rqr = \frac{\sum_{i \in S} (p(i) - a(i))^2}{\sum_{i \in S} (a(i) - \bar{a})^2}$$

dove  $\bar{a} = \frac{1}{n} \sum_{i \in S} a(i)$

e l'**errore assoluto relativo**

$$rar = \frac{\sum_{i \in S} |p(i) - a(i)|}{\sum_{i \in S} |a(i) - \bar{a}|}$$

# Il coefficiente di correlazione

- Misura la correlazione statistica tra il valore reale e quello predetto

$$corr = \frac{\sigma_{PA}}{\sigma_P \sigma_A}$$

- Caratteristiche del coefficiente di correlazione:
  - assume valori tra -1 e +1
    - valori elevati sono indice di buone prestazioni
  - è indipendente dalla scala
  - se moltiplico tutti i  $p(i)$  o tutti gli  $a(i)$  per un certo fattore  $k$ , il risultato non cambia

# Quale misura di errore usare?

- Dipende dalla situazione
- In molti casi, comunque, il metodo di classificazione migliore rimane migliore indipendentemente dal tipo di misura di errore utilizzata

	A	B	C	D
Root mean-squared error	67.8	91.7	63.3	57.4
Mean absolute error	41.3	38.5	33.4	29.2
Root relative squared error	42.2%	57.2%	39.4%	35.8%
Relative absolute error	43.1%	40.1%	34.8%	30.4%
Correlation coefficient	0.88	0.88	0.89	0.91

# Combinazione di classificatori

# Boosting e Bagging

- Sono due metodi standard per **combinare** dei classificatori  $C_1, \dots, C_T$  per produrre un classificatore  $C^*$  più accurato.
- Analogia con i medici
  - Supponiamo di voler diagnosticare una malattia. Possiamo rivolgerci a vari medici invece che ad uno solo.
  - **Bagging (bootstrap aggregating)**: prendo le risposte di tutti i medici e considero come diagnosi valida quella prodotta in maggioranza.
  - **Boosting**: peso la diagnosi di ogni medico in base agli errori che egli ha commesso in precedenza

# Bagging

- Sia  $S$  un insieme di  $s$  istanze.
- Per ogni  $t$  in  $\{1, \dots, T\}$ 
  - determino l'insieme  $S_t$  ottenuto campionando  $s$  valori con rimpiazzo
  - ottengo un classificatore  $C_t$  addestrando il mio algoritmo con l'insieme  $S_t$
- Per una nuova istanza, provo tutti i classificatori e scelgo la risposta che occorre con più frequenza.
- Lo posso usare anche per la predizione:
  - faccio la media dei valori dei vari classificatori invece della scelta a maggioranza.
  - si dimostra che per la predizione il bagging migliora sempre l'accuratezza dei risultati.



# Boosting

- Sia  $S$  un insieme di  $s$  istanze.
- Si applica ad algoritmi di classificazione che supportano istanze pesate
  - Ogni istanza ha un peso  $w$  (inizialmente uguale per tutte)
- Per ogni  $t$  in  $\{1, \dots, T\}$ 
  - addestro il classificatore  $C_t$  tenendo conto dei pesi attuali.
  - modifico i pesi in modo tale che, durante l'apprendimento di  $C_{t+1}$ , contino di più le istanze classificate incorrettamente da  $C_t$ .
- Per una nuova istanza, uso tutti i classificatori e peso i risultati in base alla loro accuratezza sull'insieme di addestramento.
- Un algoritmo vero: **Adaboost**

# Bibliografia

# Bibliografia

- Tom M. Mitchell, *Machine Learning*, McGraw-Hill
  - capitolo 5
- Ian H. Witten, Eibe Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann
  - capitolo 5, sezione 7.4
- Jean Carletta. *Assessing agreement on classification tasks: the kappa statistic*. *Computational Linguistics* 22.