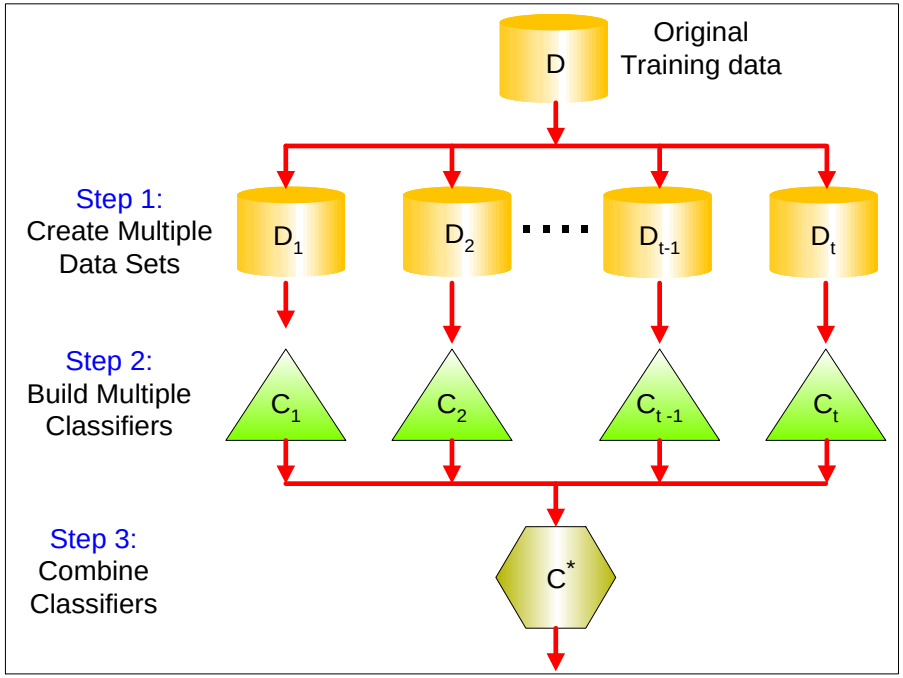
Multi classificatori

Prof. Matteo Golfarelli
Alma Mater Studiorum - Università di Bologna

Combinazione di classificatori

- Idea: costruire più classificatori di base e predire la classe di appartenza di un record aggregando le classificazioni ottenute
  - ✓ Il risultato del classificatore composto è definito per mezzo di una funzione che, per esempio, assegna il record alla classe che è stata "*votata*" dal maggior numero di classificatori

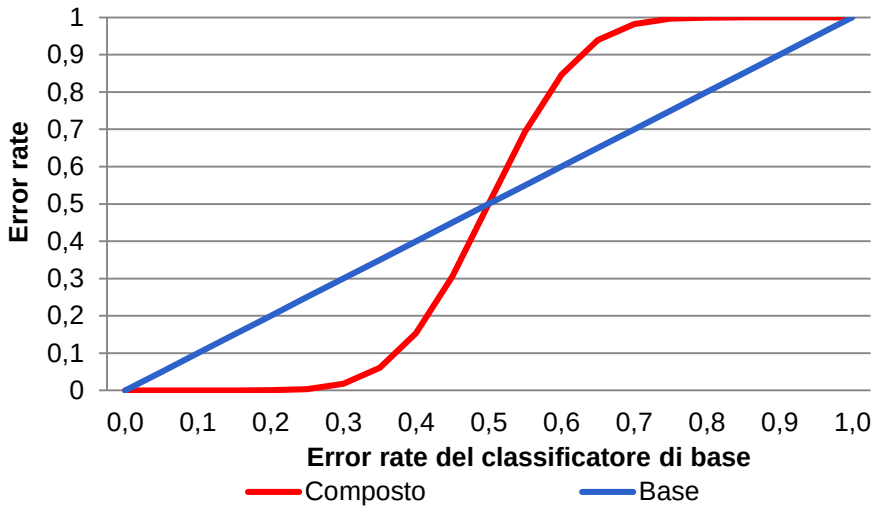


ML

Principio di funzionamento

- Supponiamo di avere 25 classificatori semplici
  - ✓ Ogni classificatore ha un error-rate pari a $\varepsilon = 0.35$
  - ✓ Si assuma che i classificatori siamo indipendenti
    - Non c'è correlazione tra gli error-rate dei classificatori
- La probabilità che il classificatore composto dia un risultato errato è:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$



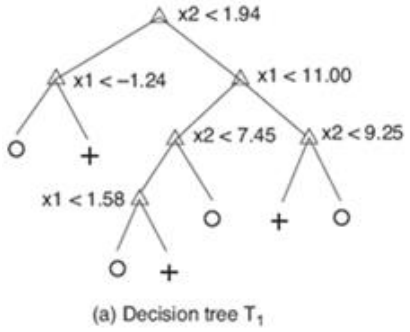
- Condizioni necessarie perchè il classificatore composto dia risultati migliori dei classificatori semplici sono:
  - ✓ Che i classificatori siano indipendenti
  - ✓ Che l'error-rate del singolo classificatore sia inferiore a 0.5

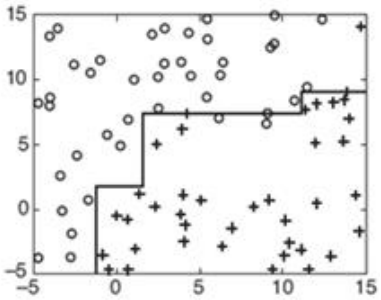
Come costruire classificatori composti

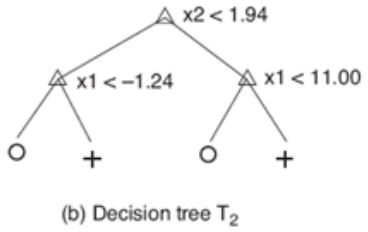
- Cambiando il training set: si costruiscono più training set a partire dal data set dato
  - ✓ Bagging e Boosting
- Cambiando gli attributi utilizzati: i singoli classificatori sono basati su un sottoinsieme degli attributi
  - ✓ Utile quando gli attributi sono fortemente ridondanti
    - Random Forest
- Cambiando le classi considerate:
  - ✓ Si partizionano le classi in due gruppi A0 e A1 e si trasforma il problema dato in un problema binario. Le classi che appartengono ad A0 sono classificate come 0 le rimanenti come 1.
  - ✓ I diversi classificatori sono costruiti risuddividendo le classi in sottoinsiemi diversi
  - ✓ La classificazione del classificatore composto si ottiene incrementando di 1 il punteggio delle classi che appartengono al sottoinsieme scelto.
  - ✓ Il record è infine assegnato alla classe che ottiene il punteggio maggiore
    - Error-Correcting Output Coding
- Cambiando i parametri dell'algoritmo di learning:
  - ✓ Topologia e pesi di una rete neurale
  - ✓ Alberi decisionali con politiche di scelta random degli attributi da utilizzare

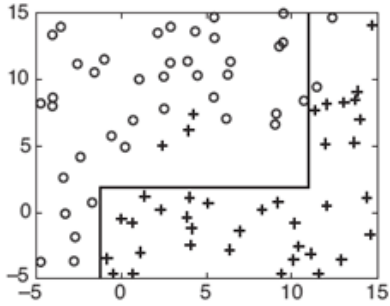
Decomposizione Bias-Variance-Noise

- Un modello formale per analizzare l'errore commesso da un classificatore
  - ✓ Probabilità che un classificatore sbagli la sua previsione
- L'errore commesso da un classificatore dipende da:
  - ✓ **BIAS:** capacità del classificatore scelto nel modellare gli eventi e nell'estendere la previsione agli eventi non presenti nel training set
    - Tipi di classificatori diversi hanno capacità diverse nel definire i decision boundary tra le classi
    - Per esempio decision tree diversi possono avere capacità diverse





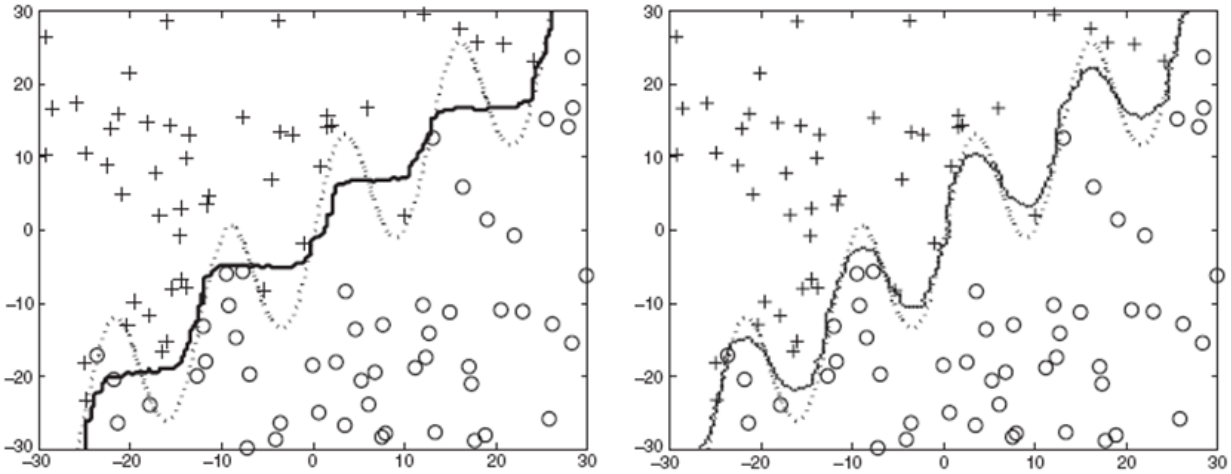




  - ✓ **VARIANCE:** capacità del training set nel rappresentare il data set reale
    - Training set diversi possono determinare decision boundary diversi
  - ✓ **NOISE:** non-determinismo delle delle classi da determinare
    - Istanze set con gli stessi valori degli attributi possono determinare classi diverse

Decomposizione Bias-Variance-Noise

- Tipi di classificatori diversi hanno capacità intrinsecamente diverse nel modellare i margini delle regioni
  - ✓ 100 training set ognuno contenente 100 esempi ottenuti a partire da una suddivisione in regioni predefinita (linea trattegiata)
  - ✓ La linea nera rappresenta la media della linea di separazione media ottenuta dai 100 classificatori



- La differenza tra la vera linea di separazione e quella media rappresenta il bias del classificatore
  - ✓ Il bias del 1-NN è inferiore
  - ✓ Tuttavia i k-NN sono maggiormente sensibili alla composizione del training set e quindi presenteranno un maggiore variance

Multi-classificatori

- Si utilizzano classificatori diversi (es. Alberi decisionali + k-nearest neighbor) per ridurre il **bias** dell'errore
  - ✓ I classificatori devono essere inidipendenti: non c'è correlazione (o ce ne è poca) tra gli errori commessi tra due classificatori
  - ✓ Classificatori diversi possono operare su sottoinsiemi distinti di attributi su cui hanno performance ideali

- La classe di appartenenza è decisa mediante un meccanismo di voting
  - ✓ Classe votata dal maggior numero di classificatori
  - ✓ Il voto può essere pesato in base alla confidenza del classificatori nel caso in cui questo la fornisca
    - Es. Es il classificatore C1 vota per la classe X. In fase di addestramento C1 ha commesso 25 su 100 errori per la classe X. Il classificatore C2 vota per la classe Y. In fase di addestramento C2 ha commesso 10 su 100 errori per la classe Y.

Il record è assegnato alla classe Y

Bagging

- Permette di costruire classificatori composti che associano un evento alla classe più votata dai classificatori base
- Ogni classificatore è costruito mediante bootstrap di un medesimo trainining set

```
// k = numero di cicli di bootstrap N = card. del training set
// δ()=1 se l'argomento della funzione è TRUE, 0 altrimenti

for i=1 to k do
 Crea un training set D_i di dimensione N
 Allena un classificatore C_i sul training set D_i
end for
```

$$C^*(\mathbf{x}) = \arg\max_y \sum_i \delta\left(C_i(\mathbf{x}) = y\right)$$

- Il bagging migliora l'errore di generalizzazione riducendo la componente **variance**
  - ✓ Il bagging sarà quindi particolarmente utile per quei tipi di classificatori sensibili alle variazioni del training set

Bagging: un esempio

- Classificatore di base: albero decisionale binario a un livello
  - ✓ Può solo fare scelte del tipo $x \leq s \rightarrow -1$ $x > s \rightarrow 1$ dove s è lo split point

- Il data set

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- L'accuratezza del classificatore base non può superare 70%
  - ✓ $x \leq 0.3 \rightarrow 1$ $x > 0.3 \rightarrow -1$
  - ✓ $x \leq 0.7 \rightarrow -1$ $x > 0.7 \rightarrow 1$

Bagging: le sessioni

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

x <= 0.65 ==> y = 1
x > 0.65 ==> y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ==> y = 1
x > 0.3 ==> y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ==> y = -1
x > 0.05 ==> y = 1

Bagging: i risultati

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True Class	1	1	1	-1	-1	-1	-1	1	1	1

- Il bagging permette di ottenere il comportamento di un albero decisionale a due livelli

Disegnare l'albero decisionale a due livelli corrispondente al risultato del bagging

Boosting

- Un approccio iterativo che permette di adattare progressivamente la composizione del training set al fine di concentrarsi sui record classificati in modo incorretto
  - ✓ Inizialmente, tutti gli N record hanno lo stesso peso (1/N)
  - ✓ Diversamente dal bagging, i pesi possono cambiare alla fine del round di boosting in modo da aumentare la probailità del record di essere selezionato nel training set
    - Vengono aumentate le probabilità dei record che sono difficili da classificare ossia che nel precedente round di boosting sono stati classificati in modo incorretto
  - ✓ Il risultato finale si ottiene combinando il risultato le predizioni fatte dai diversi classificatori
- Le tecniche di boosting si differenziano in base a come:
  - ✓ sono aggiornati i pesi dei record del training set
  - ✓ sono combinate le predizioni dei classificatori
- AdaBoost è una delle tecniche di boosting più utilizzate

AdaBoost

- Siano C_1, C_2, …, C_T i T classificatori di base ognuno utilizzato a un ciclo di boost $j \in [1,T]$. Sia ε_j l'error rate:

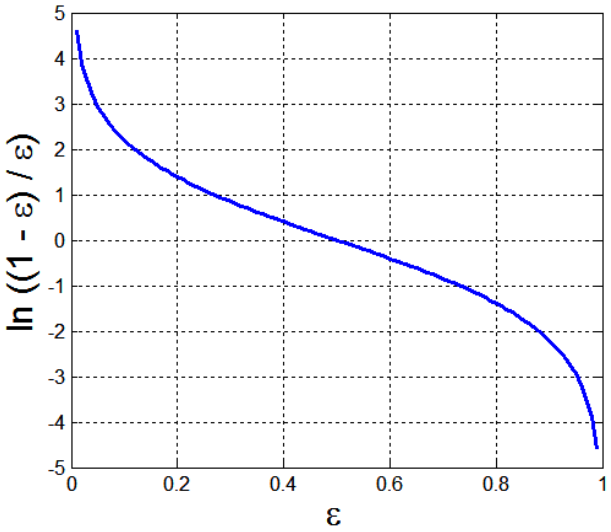
$$\varepsilon_j = \frac{1}{N}\sum_{i=1}^{N} w_i \delta\left(C_j(\mathbf{x}_i) \neq y_i\right)$$

  - ✓ $(\mathbf{x}_i, y_i)$ i=1..N record del training set
  - ✓ w_i è il peso del i-esimo elemento del training set
  - ✓ $\delta()$=1 se l'argomento è TRUE, 0 altrimenti

- L'importanza di un classificatore è definita come:

$$\alpha_j = \frac{1}{2}\ln\left(\frac{1-\varepsilon_j}{\varepsilon_j}\right)$$

- α_j assume valori positivi quando l'error rate è prossimo a 0
- α_j assume valori negativi quando l'error rate è prossimo a 1

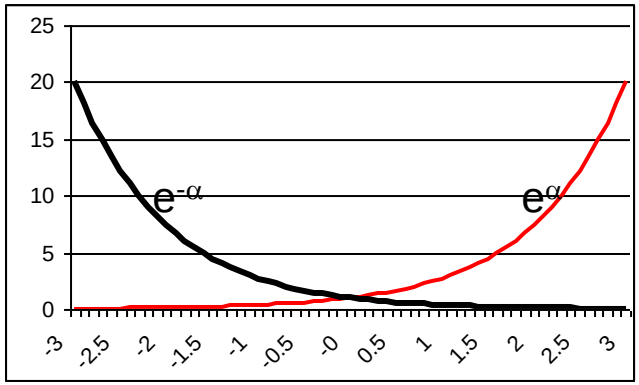


AdaBoost

- La regola per l'aggiornamento dei pesi del record *i* ad ogni ciclo di boost *j* è:

$$w_i^{(j+1)} = \frac{w_i^j}{Z_j} \begin{cases} e^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ e^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

  - ✓ Z_j è un fattore di normalizzazione per assicurare che $\sum_i w_i^{j+1} = 1$
  - ✓ Il peso dei record classificati in modo corretto si riduce, quello dei pesi classificati in modo incorretto aumenta



- Se un ciclo di boost produce un classificatore con error rate maggiore di 50%, i pesi sono riportati 1/n
- Il record è assegnato alla classe che massimizza la somma pesata:

$$C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \text{sign}\left(C_j(x) = y\right)$$

AdaBoost

1. **w**={w_i=1/N i=1..N}
2. **for** j=1 to T **do** // numero dei cicli di boost
3. Crea un tr. set D_j campionando con replacement in base a **w**
4. Allena un classificatore C_j utilizzando D_j
5. Applica C_j a tutti gli esempi contenuti in D
6. ε_j=1/N Σ_i w_i $\delta(C_j(\mathbf{x}_i)\neq y_i)$ // Calcola l'errore pesato
7. **if** ε_j > 0.5 **then**
8. **w**={w_i=1/N|i=1,2,…,N}; // reset!
9. **else**
10. α_j=1/2 ln((1-ε_j)/ε_j);
11. Aggiorna i pesi **w**;
12. **end if;**
13. **end for;**
14. $C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \, \text{sign}(C_j(x) = y)$ // sign restituisce 1/-1 se la classificazione è corretta/sbagliata

AdaBoost: un esempio

- Classificatore di base: albero decisionale binario a un livello
  - ✓ Può solo fare scelte del tipo $x \leq s \rightarrow -1$ $x > s \rightarrow 1$ dove s è lo split point

- Il data set

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- L'accuratezza del classificatore base non può superare 70%
  - ✓ $x \leq 0.3 \rightarrow 1$ $x > 0.3 \rightarrow -1$
  - ✓ $x \leq 0.7 \rightarrow -1$ $x > 0.7 \rightarrow 1$

AdaBoost: un esempio

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	-1	-1	-1	-1	-1	-1	-1	-1	1	1

Split point: 0.75
ε_1 =0.03
α_1=1.738

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Split point: 0.05
ε_2 =0.004
α_2=2.7784

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

Split point: 0.30
ε_3 =0.00027
α_3=4.1195

(a) Training records chosen during boosting

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

(b) Weights of training records

AdaBoost: un esempio

- Il dataset

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Il risultato

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

α_1=1.738
α_2=2.7784
α_3=4.1195

- ✓ 5.16 = -1.738 + 2.7784 + 4.1195
- ✓ Il classificatore non commette errori ed ha un comportamento ottenibile con un albero a due livelli