

# Alberi di decisione per la classificazione

Note per Metodi di Ottimizzazione per *Big Data*  
A.A. 2018-19

## Un esempio introduttivo: Classificare le “mattine del sabato” in base alle condizioni meteo

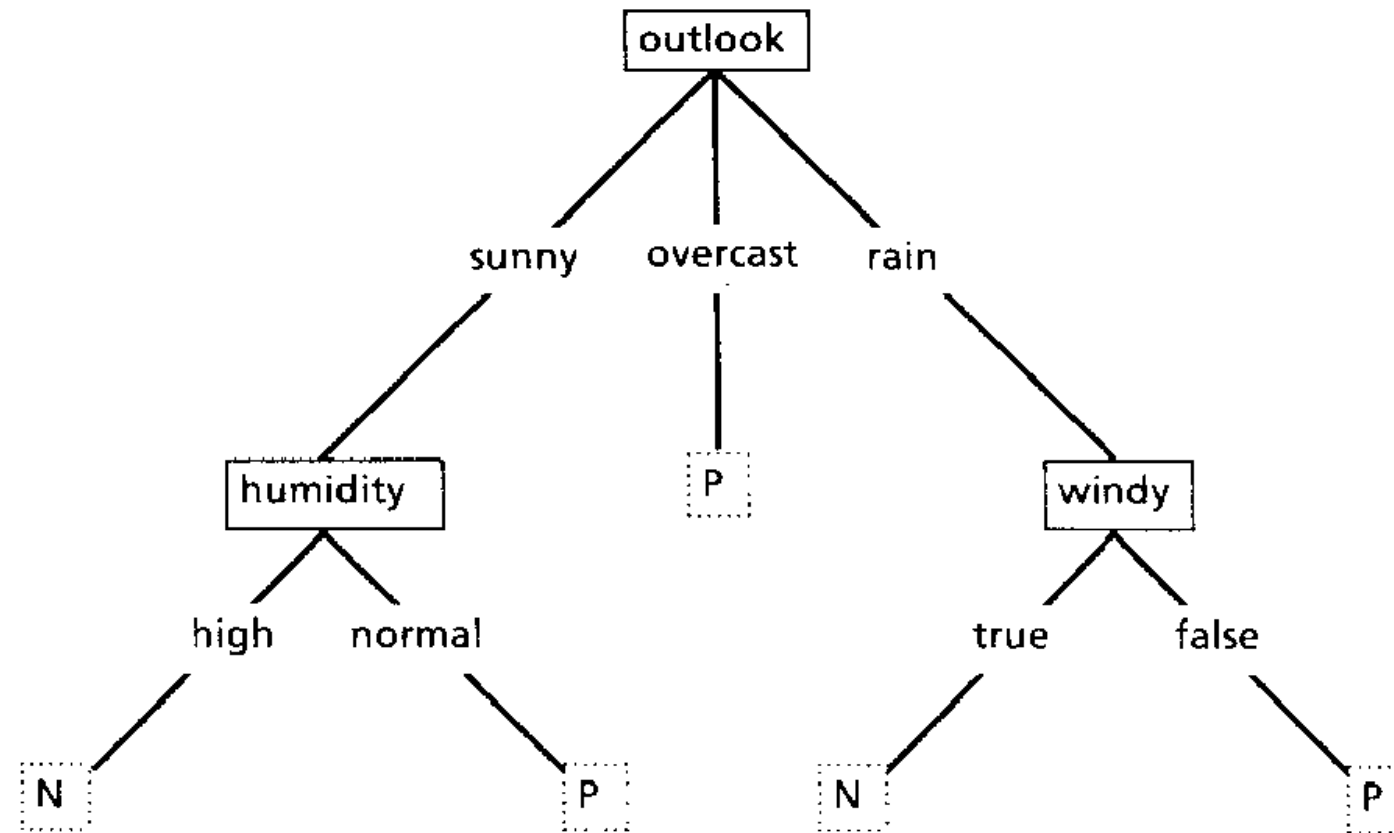
da: J.R. Quinlan. Induction of Decision Trees, *Machine Learning*, 1, 81-106, 1986

Table 1. A small training set

| No. | Attributes |             |          |       | Class |
|-----|------------|-------------|----------|-------|-------|
|     | Outlook    | Temperature | Humidity | Windy |       |
| 1   | sunny      | hot         | high     | false | N     |
| 2   | sunny      | hot         | high     | true  | N     |
| 3   | overcast   | hot         | high     | false | P     |
| 4   | rain       | mild        | high     | false | P     |
| 5   | rain       | cool        | normal   | false | P     |
| 6   | rain       | cool        | normal   | true  | N     |
| 7   | overcast   | cool        | normal   | true  | P     |
| 8   | sunny      | mild        | high     | false | N     |
| 9   | sunny      | cool        | normal   | false | P     |
| 10  | rain       | mild        | normal   | false | P     |
| 11  | sunny      | mild        | normal   | true  | P     |
| 12  | overcast   | mild        | high     | true  | P     |
| 13  | overcast   | hot         | normal   | false | P     |
| 14  | rain       | mild        | high     | true  | N     |

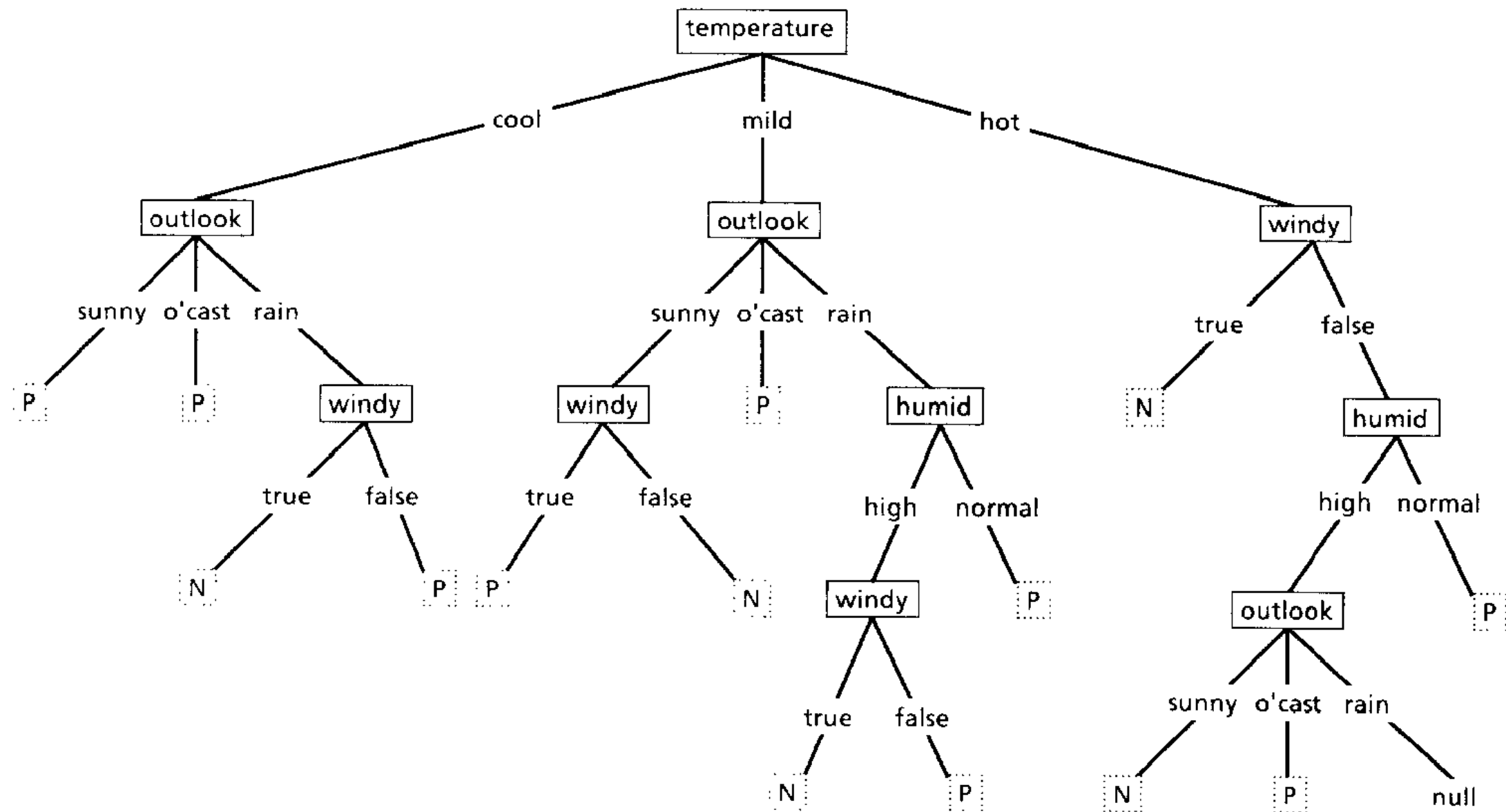
## Un albero di classificazione semplice per le “mattine del sabato”

J.R. Quinlan. Induction of Decision Trees, *Machine Learning*, 1, 81-106, 1986



## Un albero di classificazione più complesso per le “mattine del sabato”

J.R. Quinlan. Induction of Decision Trees, *Machine Learning*, 1, 81-106, 1986



# *Optimal classification trees*

**Dimitris Bertsimas & Jack Dunn**

**Machine Learning**

ISSN 0885-6125

Volume 106

Number 7

Mach Learn (2017) 106:1039-1082

DOI 10.1007/s10994-017-5633-9



 Springer

# OCT-MIO

# OCT-MIO

## Variabili

$$z_{it}, l_t \in \{0, 1\}, \quad i = 1, \dots, n, \quad \forall t \in \mathcal{T}_L,$$
$$a_{jt}, d_t \in \{0, 1\}, \quad j = 1, \dots, p, \quad \forall t \in \mathcal{T}_B.$$

$$b_t \in [0, 1], \quad t \in \mathcal{T}_B$$

## Variabili “ausiliarie”

$$c_{kt} \in \{0, 1\}, \quad L_t \in \mathbb{R}_+ \quad k = 1, \dots, K; \quad t \in \mathcal{T}_L$$

## Variabili “riassuntive”

$$N_t \in \mathbb{Z}_+, \quad N_{kt} \in \mathbb{Z}_+, \quad k = 1, \dots, K; \quad t \in \mathcal{T}_L$$

# OCT-MIO

**Vincoli: Tree structure**

$$\sum_{j=1}^p a_{jt} = d_t, \quad \forall t \in \mathcal{T}_B,$$

$$0 \leq b_t \leq d_t, \quad \forall t \in \mathcal{T}_B,$$

$$d_t \leq d_{p(t)}, \quad \forall t \in \mathcal{T}_B \setminus \{1\},$$

# OCT-MIO

## Vincoli: Allocazione (punti-foglie)

$$\sum_{t \in \mathcal{T}_L} z_{it} = 1, \quad i = 1, \dots, n,$$

$$z_{it} \leq l_t, \quad \forall t \in \mathcal{T}_L,$$

$$\sum_{i=1}^n z_{it} \geq N_{\min} l_t, \quad \forall t \in \mathcal{T}_L,$$

## Vincoli: Consistency

$$a_m^T x_i \geq \overset{b_m}{\cancel{b_t}} - (1 - z_{it}), \quad i = 1, \dots, n, \quad \forall t \in \overset{\mathcal{T}_L}{\cancel{\mathcal{T}_B}}, \quad \forall m \in A_R(t),$$

$$a_m^T (x_i + \epsilon) \leq \underset{b_m}{\cancel{b_t}} + (1 + \epsilon_{\max})(1 - z_{it}), \quad i = 1, \dots, n, \quad \forall t \in \underset{\mathcal{T}_L}{\cancel{\mathcal{T}_B}}, \quad \forall m \in A_L(t),$$



# OCT-MIO

## Vincoli: Misclassification error

$$L_t \geq N_t - N_{kt} - n(1 - c_{kt}), \quad k = 1, \dots, K, \quad \forall t \in \mathcal{T}_L,$$

$$L_t \leq N_t - N_{kt} + nc_{kt}, \quad k = 1, \dots, K, \quad \forall t \in \mathcal{T}_L,$$

$$L_t \geq 0, \quad \forall t \in \mathcal{T}_L,$$

$$N_{kt} = \frac{1}{2} \sum_{i=1}^n (1 + Y_{ik}) z_{it}, \quad k = 1, \dots, K, \quad \forall t \in \mathcal{T}_L,$$

$$N_t = \sum_{i=1}^n z_{it}, \quad \forall t \in \mathcal{T}_L,$$

$$\sum_{k=1}^K c_{kt} = l_t, \quad \forall t \in \mathcal{T}_L,$$

← Vincoli non necessari.  
Perché?

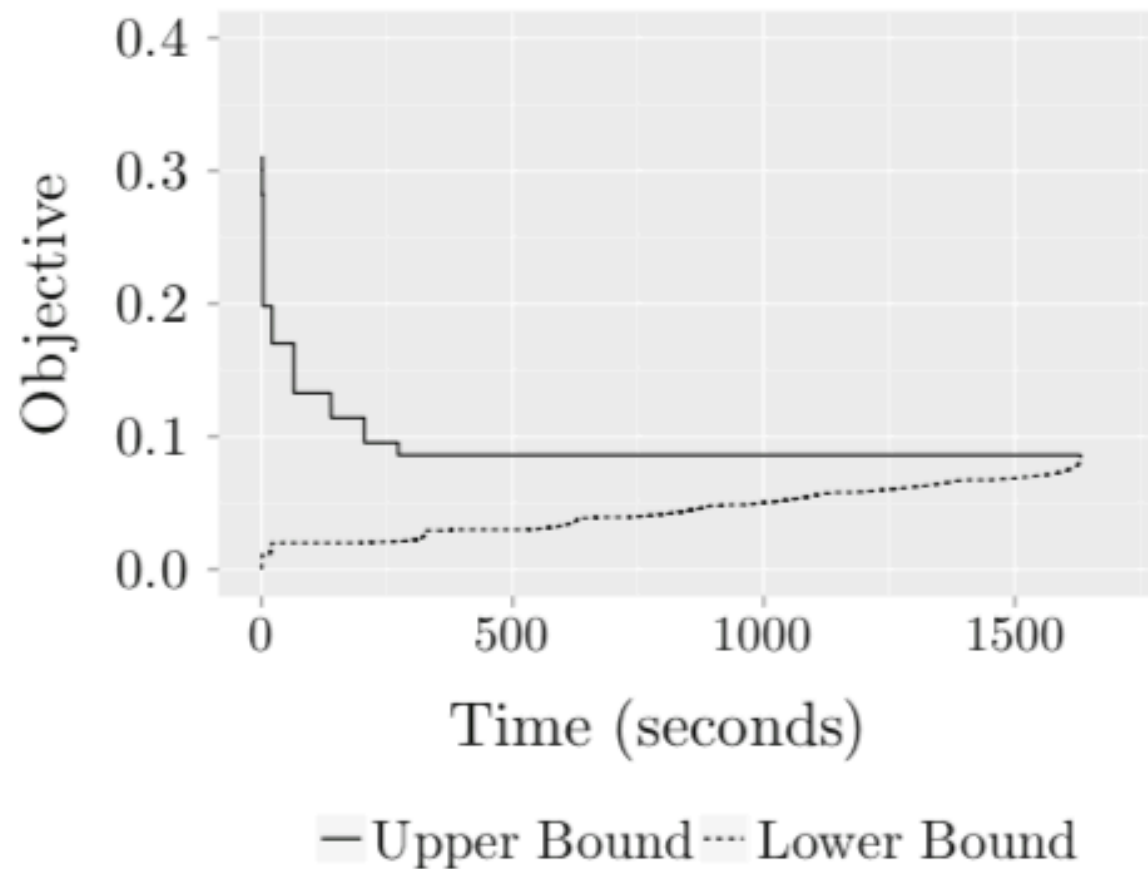
Perché  $c_{kt}$  assume  
il val. corretto?

# OCT-MIO

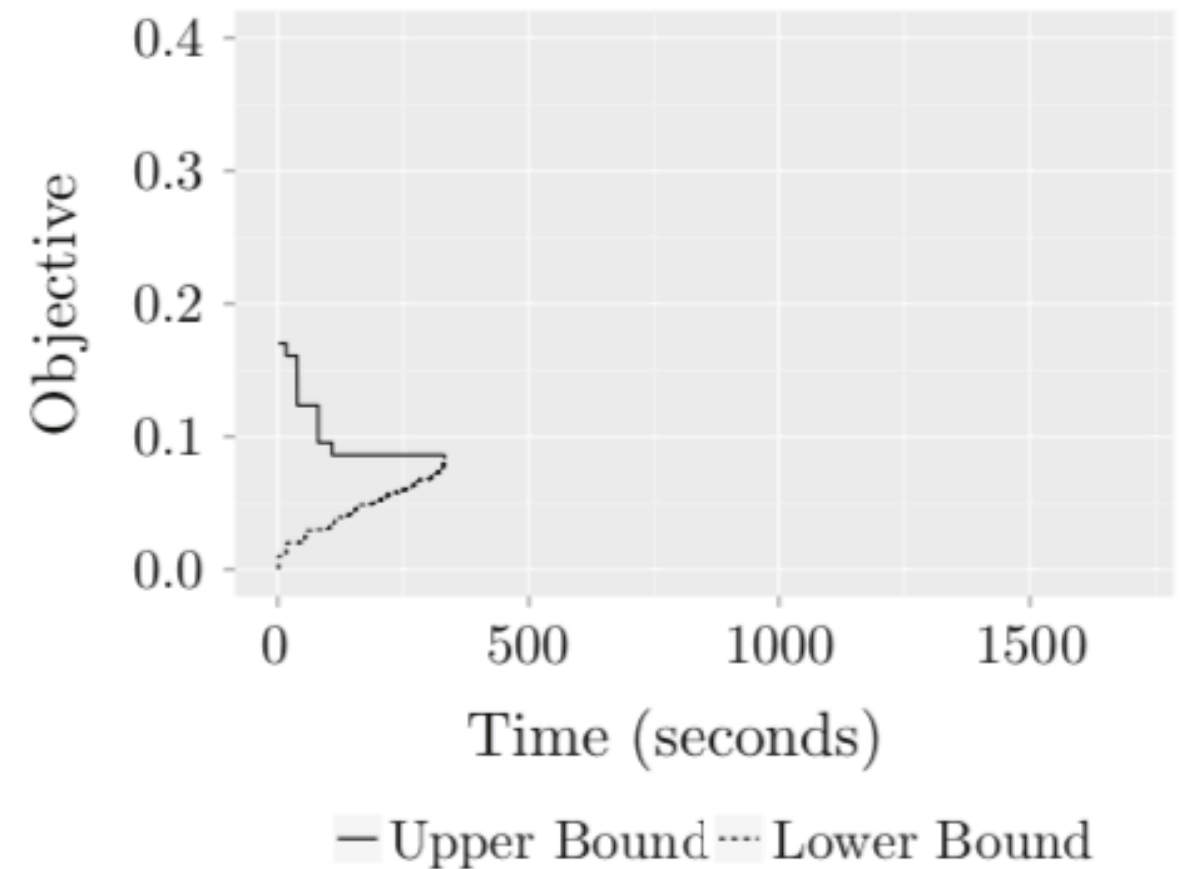
## Obiettivo

$$\min \quad \frac{1}{\hat{L}} \sum_{t \in \mathcal{T}_L} L_t + \alpha \sum_{t \in \mathcal{T}_B} d_t.$$

# Warm start



**(a)**



**(b)**

**Fig. 3** Comparison of *upper* and *lower* bound evolution while solving MIO problem (24) with and without warm starts for a tree of depth  $D = 2$  for the Wine dataset with  $n = 178$  and  $p = 13$ . **a** Without warm start, **b** with warm start

# Warm start

- La soluzione iniziale (warm start) fornita da un alg. greedy (CART, ID3, C4.5) può essere lontana dall'ottimo
- La maggior parte del tempo di calcolo serve a **provare** che la sol. corrente è un ottimo, **non a determinarlo**



idea per **euristica**: early stop

- Soluzioni del MIO a profondità  $D$  possono essere usate come warm start per il modello a  $D+1$

# Addestramento di un OCT

Scelta degli iper-parametri

- **Profondità:** scegli una profondità max  $D_{\max}$ , genera i DT a partire da  $D = 2$  fino a  $D_{\max}$ , usando le soluzioni come warm start per i problemi a  $D$  maggiore (*pool* di soluzioni warm start)
- **Parametro di complessità  $\alpha$ :** portare il termine di complessità nei vincoli

**Pb(C):**

$$\sum_{t \in \mathcal{T}_B} d_t \leq C$$

Ricerca su  $C = 1, \dots, C_{\max} = 2^D - 1$  (max numero split). La soluzione con  $C = k$  è feasible (warm start) con  $C = k+1$

- I valori di  $\alpha$  che rendono le soluzioni del problema Pb(C) ottime per OCT-MIO costituiscono i candidati per la scelta del parametro di complessità

# Addestramento di un classificatore ad albero di decisione da OCT-MIO

1. Scegli profondità max  $D_{\max}$  e min leaf size  $N_{\min}$
2. **For**  $D = 1, \dots, D_{\max}$  **do**:
  - **For**  $C = 1, \dots, 2^D - 1$  **do**:
    - A. **Run** CART con  $\alpha = 0$  e  $N_{\min}$ . Prune a profondità  $D$ , max num. split =  $C$ . Inserisci la sol. nel pool di warm start
    - B. Scegli candidato più accurato (su validation set) nel pool di warm start
    - C. Risolvi  $P_b(C)$  con profondità  $D$  e  $C$  split usando il warm start selezionato. Inserisci la sol. nel pool di warm start
3. Post-process: rimuovi le sol. non ottime per OCT-MIO per alcun valore di  $\alpha$
4. Seleziona le sol. migliori sul validation set. Determina range per  $\alpha$

# Addestramento di un classificatore ad albero di decisione da OCT-MIO

1. Scegli profondità max  $D_{\max}$ , min e **max** leaf size  $N_{\min}$  e  $N_{\max}$
2. **For size =  $N_{\max}$  downto  $N_{\min}$  do:**
  - **For  $D = 1, \dots, D_{\max}$  do:**
    - **For  $C = 1, \dots, 2^D - 1$  do:**
      - A. **Run** CART con  $\alpha = 0$  e **size**. Prune a profondità  $D$ , max num. split =  $C$ . Inserisci la sol. nel pool di warm start
      - B. Scegli candidato più accurato (su validation set) nel pool di warm start
      - C. Risolvi  $P_b(C)$  con profondità  $D$  e  $C$  split usando il warm start selezionato. Inserisci la sol. nel pool di warm start
3. Post-process: rimuovi le sol. non ottimali per OCT-MIO per qualche  $\alpha$
4. Seleziona le sol. migliori sul validation set. Determina range per  $\alpha$

# Adattamento di OCT-MIO al caso di alberi di decisione **multivariati**

...



# Pruning

- Alberi complessi (che danno ottimi risultati sul training set) soggetti a **overfit**
- Alberi più semplici/piccoli (meno split) possono
  - essere più interpretabili
  - produrre risultati con minore varianza (al costo di maggior bias)
- Una strategia: costruire albero complesso e ricavare un sottoalbero significativo attraverso il *taglio di alcuni rami/ sottoalberi* (**pruning**)

# Alberi di decisione: Pro & Con.s

- Semplici da spiegare/giustificare ai non esperti. Sono illustrati graficamente e facilmente interpretabili
- Rappresentano in modo più verosimile il processo decisionale umano rispetto ad altri metodi di regressione e classificazione
- Gli alberi modellano in modo più immediato variabili qualitative
- Livello di **accuratezza inferiore** rispetto ad altri metodi di regressione e classificazione
- **Poco “robusti”**: piccole variazioni nei dati producono significativi cambiamenti nell'albero prodotto


# Bagging, Random Forests, Boosting

- DT soffrono di **varianza elevata**:
  - estrai in modo casuale due (o più) dataset da una popolazione
  - ricava alberi di decisione sulla base dei nuovi dataset
  - alberi di decisione (probabilmente) molto dissimili
- Bootstrap aggregation (**bagging**): metodo *general-purpose* per ridurre la varianza – aumentando l'accuratezza della predizione – aggregando un insieme di osservazioni
- Idealmente, potrei
  - ricavare un certo numero  $B$  di training set dalla popolazione osservata
  - considerare la media delle  $B$  predizioni come predizione del modello aggregato
- Tipicamente, non si ha disponibilità di molteplici training set

# Bagging

- **Bootstrap:**
  - ricavare  $B$  campioni con reimmissione (estrazione bernouilliana) dal training set, ottenendo dei (bootstrap) training set  $1, \dots, B$
- **Bagging** (regressione - dati quantitativi)
  - **For**  $b = 1, \dots, B$  **do**
    - addestra il sistema sui dati del training set bootstrap  $b$
    - ricava una predizione  $f^b(x)$
  - **Output** predizione media (  $1/B \sum_b f^b(x)$  )

# Bagging

- Applicare questa idea agli alberi di classificazione (dati qualitativi)
- Costruire un albero di classificazione a partire da ciascuno dei training set bootstrap  $1, \dots, B$
- Classificare sulla base del “voto di maggioranza”: la predizione è la classe più frequente tra i risultati ottenuti dai  $B$  alberi
- Non viene effettuato pruning degli alberi generati a partire dai training set bootstrap (  alberi con alta varianza ma basso *bias*). La varianza viene ridotta prendendo il risultato “a maggioranza” sulle  $B$  classificazioni

# Stima dell'errore “Out-Of-Bag”

- Tecnica di stima dell'errore che riduce il costo computazionale (risp. ad altre tecniche, ad es. cross-validazione)
- Un DT nel bagging usa - in media - circa  $2/3$  dei dati originali
- $1/3$  circa delle osservazioni non sono utilizzati per generare un singolo DT: punti/osservazioni **Out-Of-Bag** (OOB)
- **Idea:** usare i risultati prodotti dai DT che hanno  $x_i$  OOB (quindi circa  $B/3$  alberi) per ottenere una predizione su  $x_i$  (voto di maggioranza sui circa  $B/3$  risultati)
- Ripetere la procedura per tutti gli  $n$  punti  $x_i$   $i = 1, \dots, n$  e calcolare l'errore di classificazione complessivo
- Stima valida (i punti del dataset che non sono stati utilizzati per la generazione degli alberi di cui si considera il risultato, vengono usati come punti di un validation set)

# Bagging


- Bagging aumenta l'accuratezza della predizione al costo di una minore interpretabilità
- Gini index come misura dell'importanza di una variabile indipendente/feature  $j$  :
  - **Forall** albero  $T_b$  dal training set bootstrap  $b = 1, \dots, B$  :
    - Calcola diminuzione  $d_{bj}$  del Gini index associata allo split sulla variabile  $j$
  - Output (  $1/B \sum_b d_{bj}$  ) : **importanza relativa della variabile  $j$**

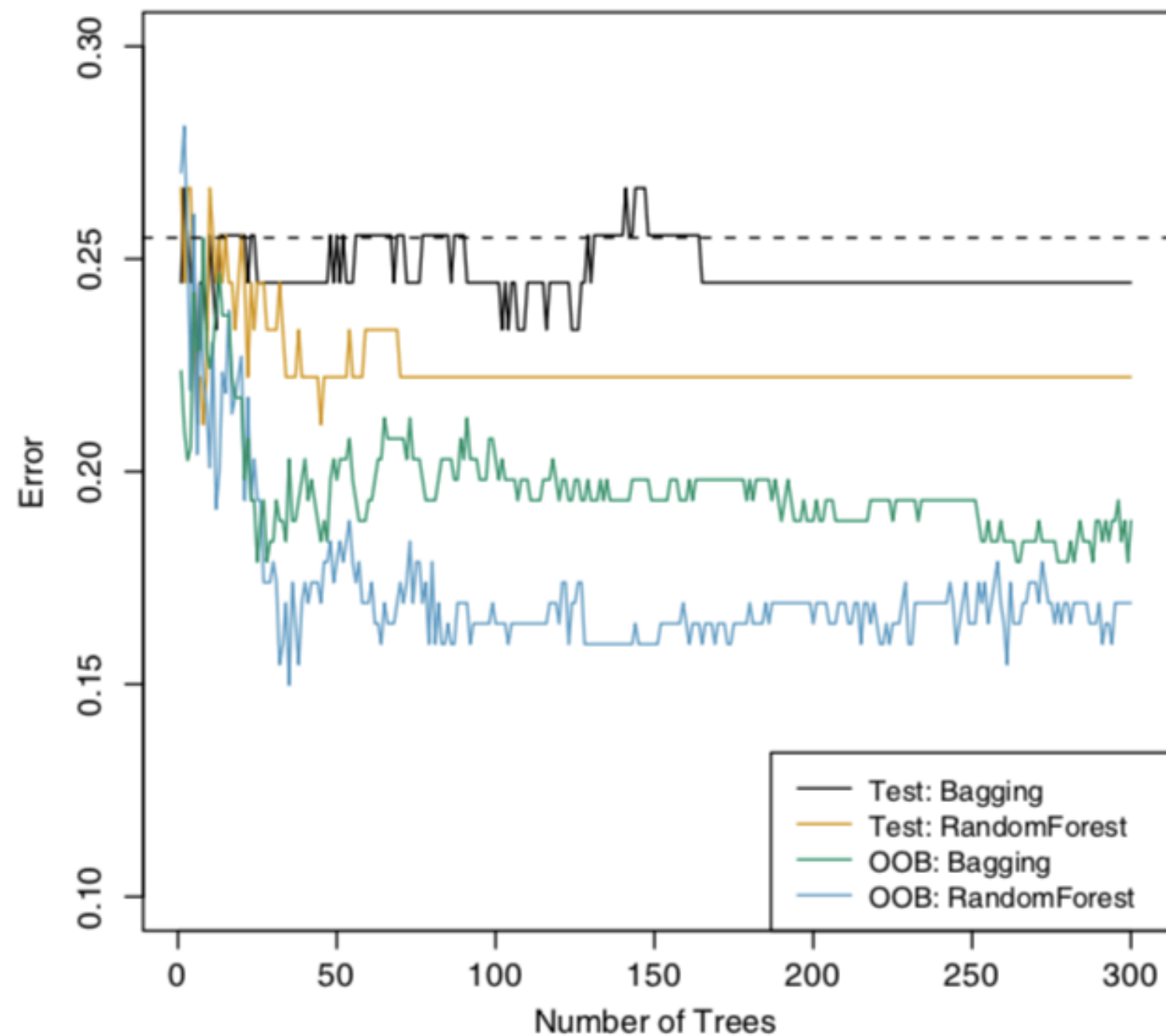
# Random Forests

- Random Forests © sono un miglioramento del bagging basato sulla scelta di alberi meno correlati tra loro
- Come nel bagging, costruiamo  $B$  alberi di decisione (a partire da  $B$  training set bootstrap)
- Nella costruzione top-down degli alberi, ad ogni split:
  - considera soltanto un **campione random** di  $m < p$  variabili candidate (solo questo sottoinsieme è candidato)
  - tipicamente si pone  $p \approx \sqrt{m}$

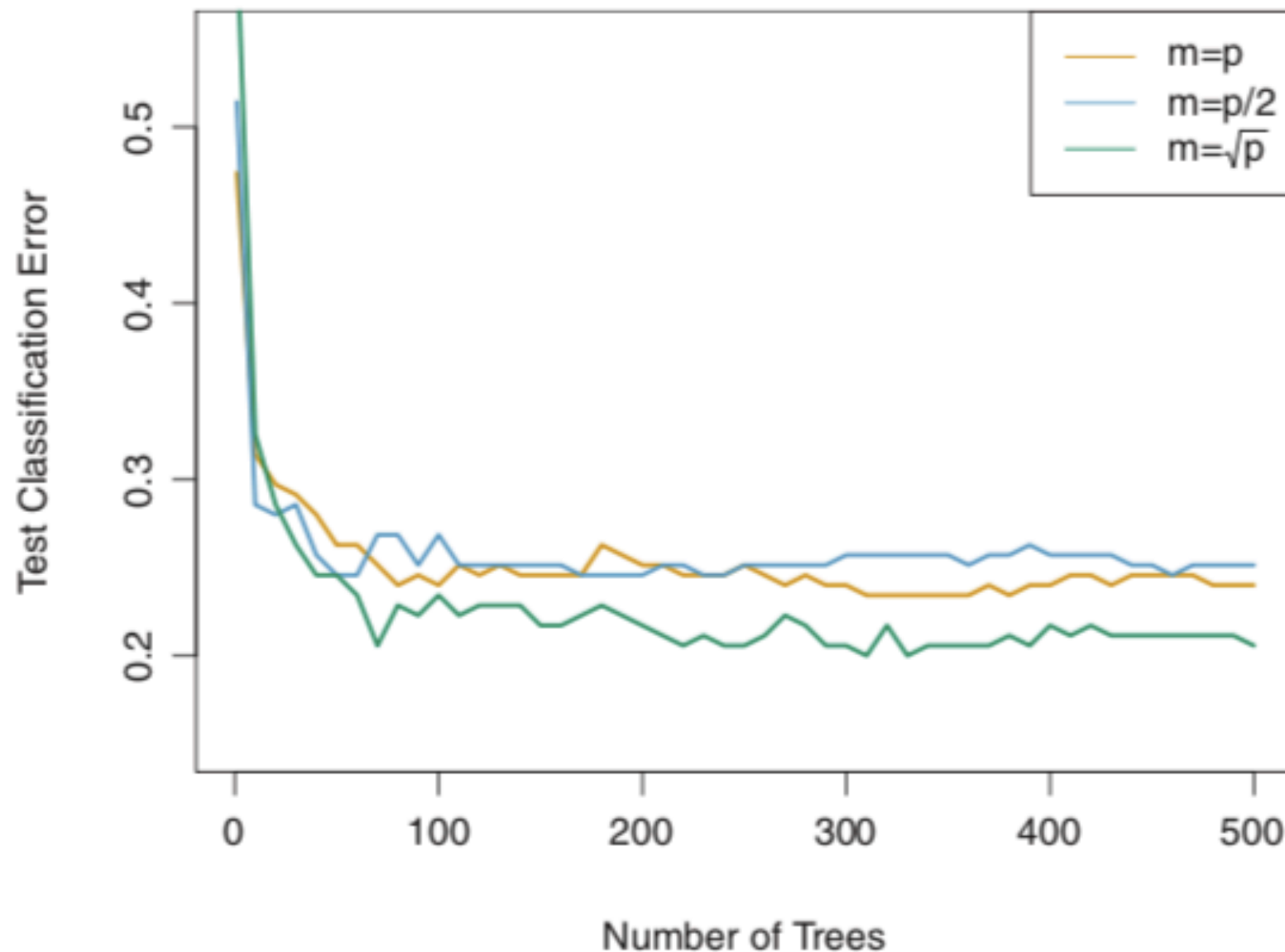


# Random Forests

- Ridurre il numero di alternative a ogni split?
  - riduce rischi di possibile forte correlazione tra alberi (p.es. in presenza di pochi *strong predictor*)
  - aumenta la riduzione della varianza promuovendo la costruzione di alberi poco correlati tra loro
- ponendo  $m = p$   random forests = bagging
- Con un numero alto di attributi fortemente correlati, scegliere  $m$  piccolo (risp. a  $p$ ) può migliorare molto le performance di random forests rispetto al bagging



Bagging e Random Forests su Patologie cardiache (James et. al. Introduction to Statistical Learning, Springer)  
 $K = 2$  classi,  $p = 13$  attributi,  $n = 303$



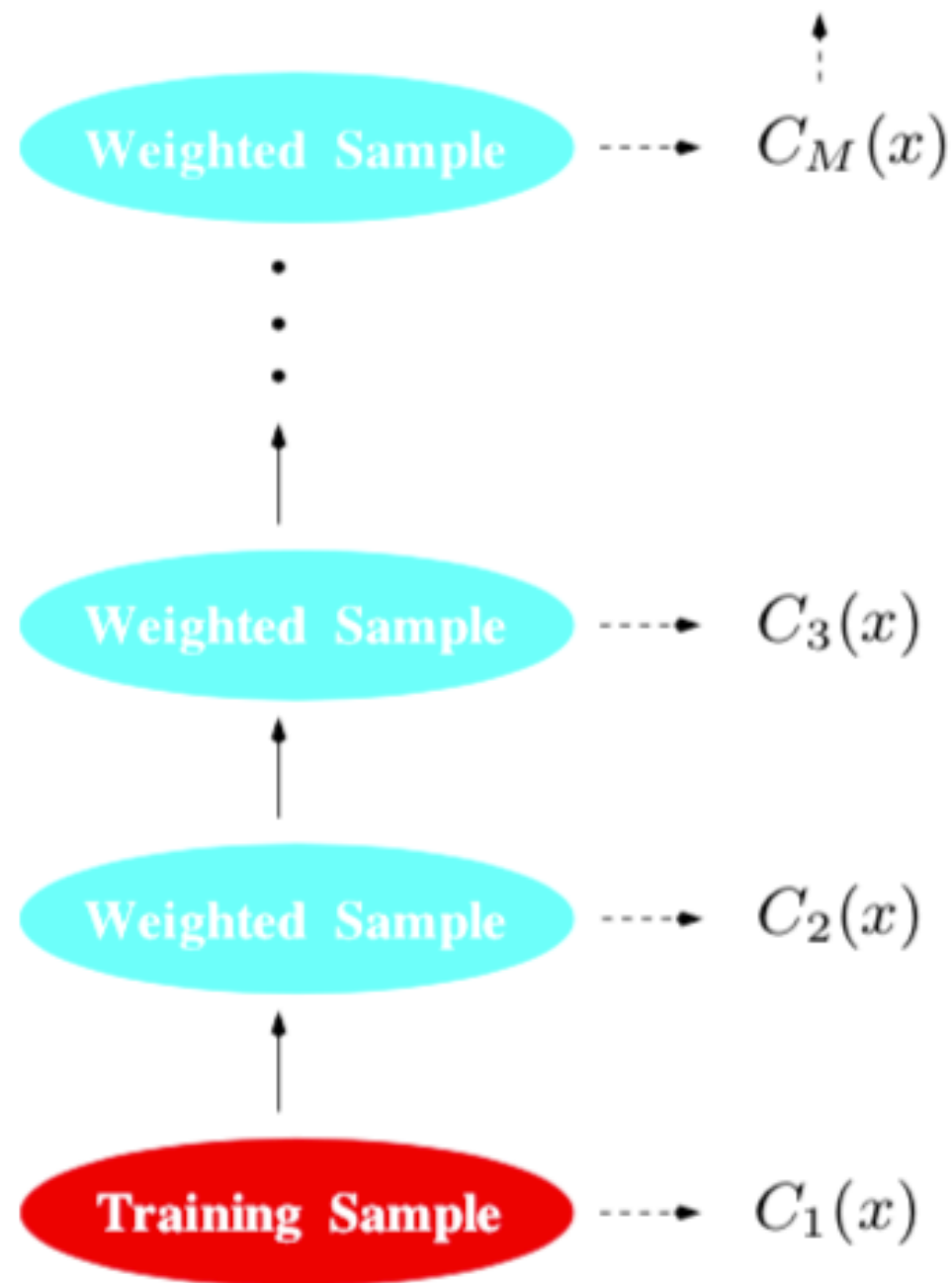
Risultati di Random Forests su Livelli di attività di geni e classi tumorali (James et. al. Introduction to Statistical Learning, Springer)

$K = 15$  classi,  $p = 500$  attributi,  $n \approx 350$  (divisi in modo casuale tra training e test set)

# Boosting

- Il **boosting** è un approccio applicabile a metodi diversi di apprendimento statistico di regressione e classificazione
- Nel bagging (e Random Forests),  $B$  alberi di decisione sono costruiti a partire da  $B$  training set bootstrap **in modo indipendente** gli uni da gli altri
- Nel boosting ogni albero viene creato a partire da informazioni sugli alberi creati in precedenza
- Anche in questo caso si tratta di combinare un numero (sufficientemente elevato)  $B$  di alberi di decisione a partire da  $B$  training set
- I training set non sono bootstrap: creati modificando opportunamente i dati originali

# Boosting



- I campioni successivi sono determinati “pesando” i dati dei campioni precedenti
- Il peso di un’osservazione  $x_i$  al passo  $b$  è maggiore (maggiore probabilità di essere ripetuto) se il DT non ha classificato correttamente  $x_i$  al passo  $(b - 1)$
- La classificazione viene effettuata pesando opportunamente i risultati dei vari DT

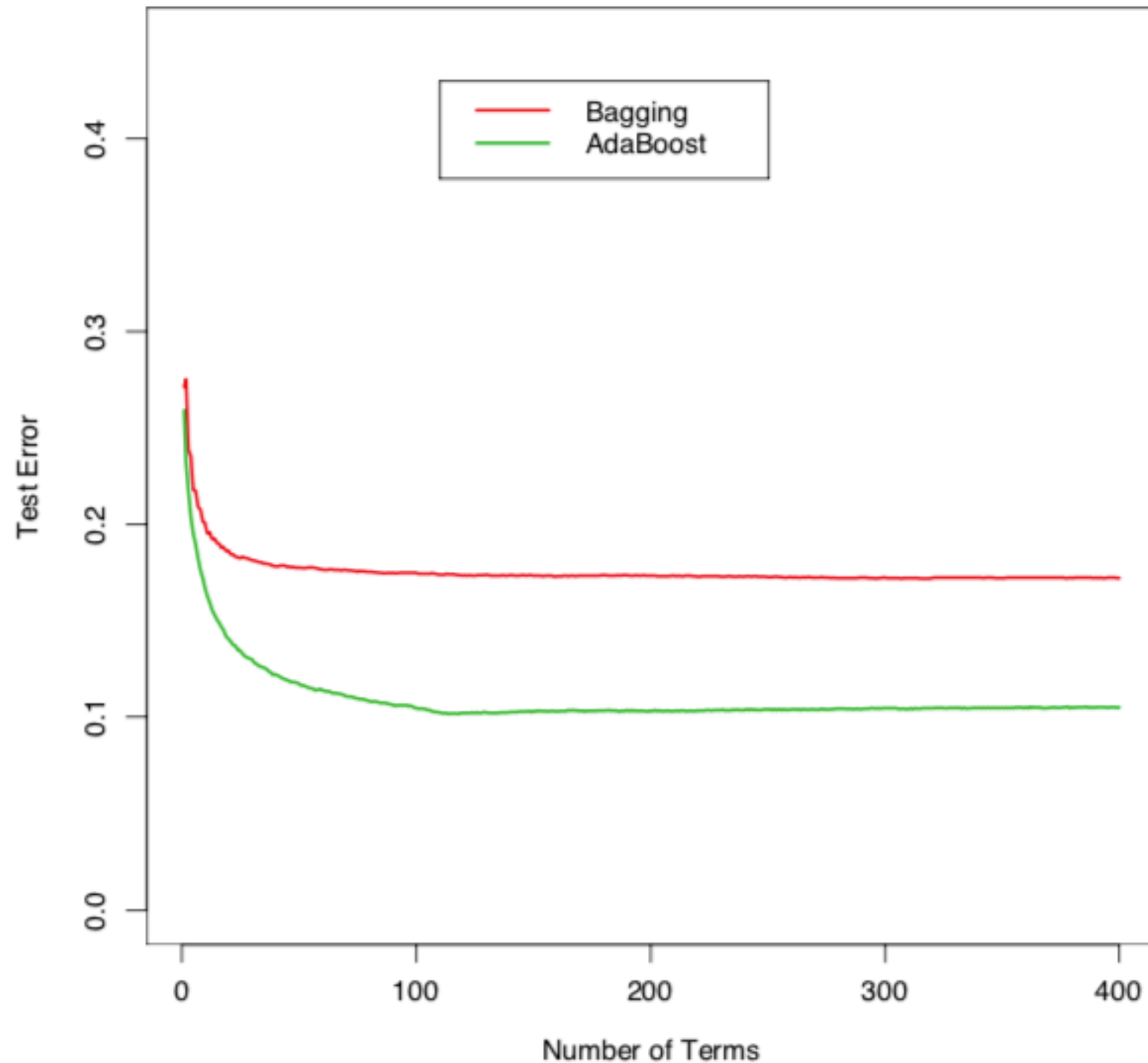
# Boosting

## AdaBoost (Freund, Shapire, 1996)

- Init. pesi dei dati  $\mathbf{x}_i$  del training set:  $w_i = (1/n)$ ,  $i = 1, \dots, n$
- **For**  $b = 1, \dots, B$  **do**:
  - Genera DT( $b$ ) su training data  $b$ -mo usando pesi  $w_i$
  - Calcola errore pesato di DT( $b$ ) :  $e(b) = \sum w_i * I(i \text{ misclass.}) / \sum w_i$
  - Calcola  $\alpha(b) = \log[(1 - e(b)) / e(b)]$
  - Aggiorna:  $w_i = w_i * \exp[\alpha(b) * I(i \text{ misclass.})]$  e rinormalizza
- Output risultato pesato con gli  $\alpha(b)$

# Boosting vs. Bagging

100 Node Trees

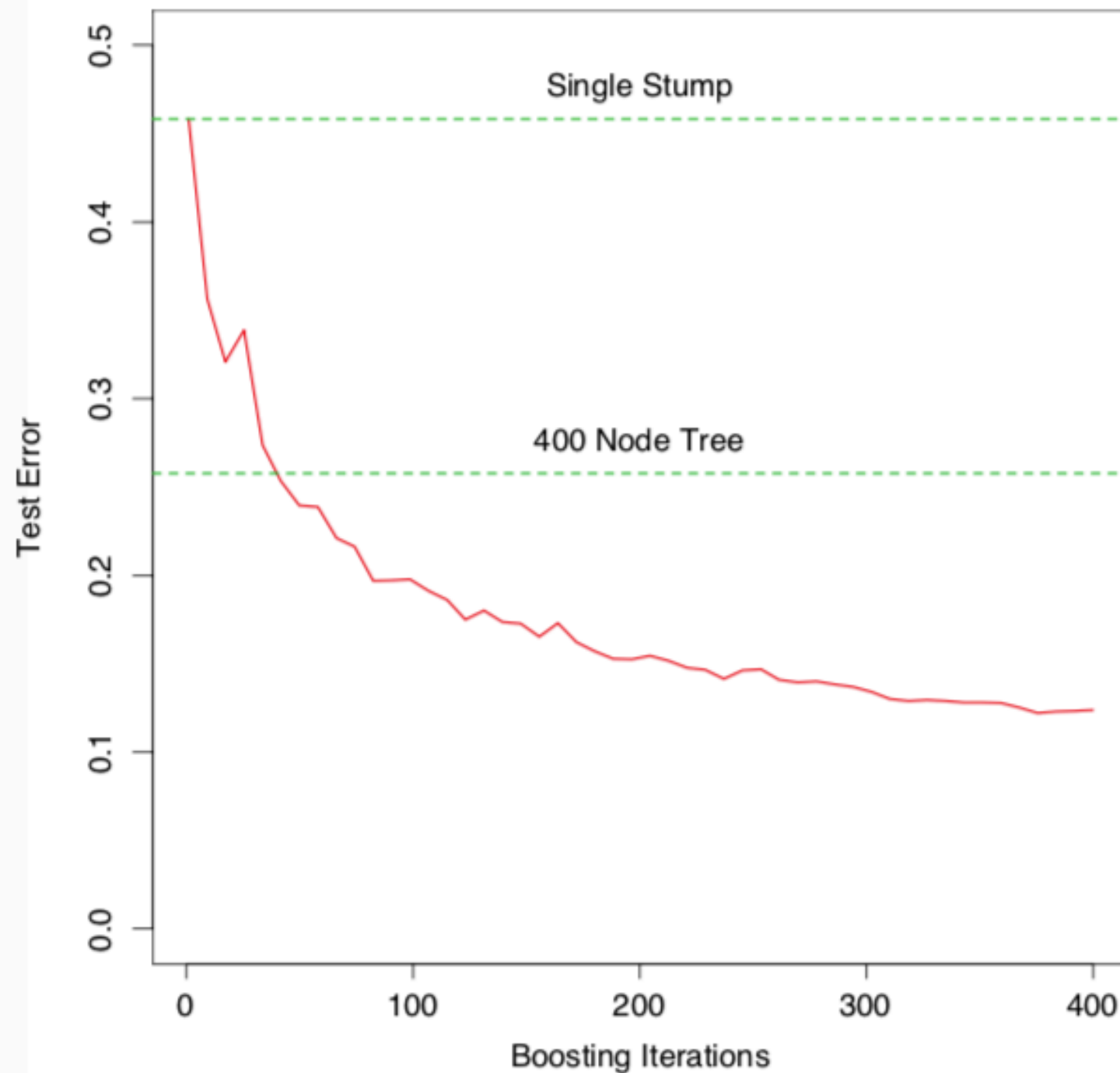


T. Hastie, Stanford U.:

Problema delle sfere annidate in dim. 10

$n = 2000$   $K = 2$

# Boosting di “stump”



T. Hastie, Stanford U.:  
Problema delle sfere annidate in dim. 10  
 $n = 2000$   $K = 2$