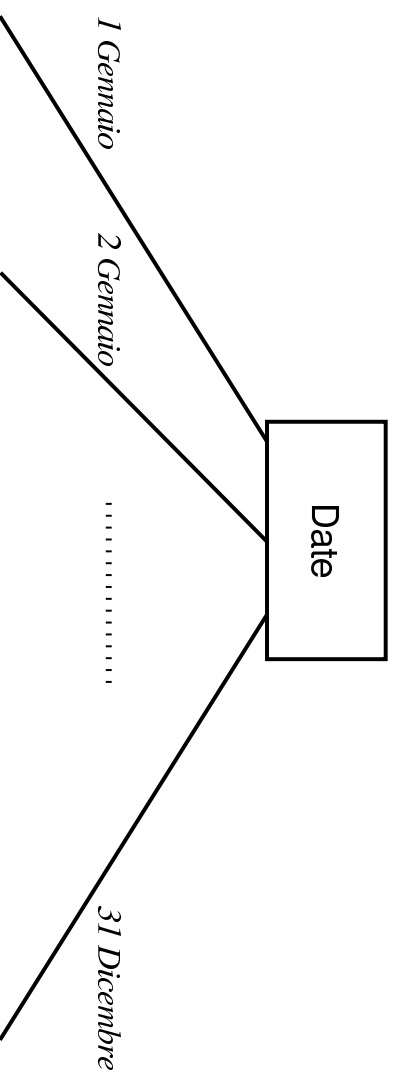


## Selezione Attributo Ottimo: Problema

**Problema:** Il Guadagno favorisce troppo attributi che possono assumere tanti valori diversi.

Esempio: se al problema di decidere quando giocare a tennis si aggiunge un attributo che consiste nella data del giorno considerato (es. Data="11 Novembre"), allora l'attributo Date è quello che avrà guadagno massimo (ogni giorno costituirà un sottoinsieme diverso e puro, quindi con entropia a 0), anche se in realtà non è significativo.



## Selezione Attributo Ottimo: Gain Ratio

Per rimediare a questo problema si definisce il *Gain Ratio*:

$$\text{GainRatio}(S, A) = \frac{\text{Guadagno}(S, A)}{\text{SplitInformation}(S, A)}$$

dove

$$\text{SplitInformation}(S, A) = - \sum_{v \in \text{Valori}(A)} \frac{|S_{A=v}|}{|S|} \log_2 \left( \frac{|S_{A=v}|}{|S|} \right)$$

La *SplitInformation* misura quanti, e quanto uniformi sono, i sottoinsiemi generati dall'attributo  $A$  a partire dall'insieme  $S$ .

*SplitInformation* corrisponde alla entropia di  $S$  dati i valori di  $A$ .

Si osservi che il termine *SplitInformation* favorisce attributi che suddividono  $S$  in molti sottoinsiemi tutti della stessa cardinalità.

## Selezione Attributo Ottimo: Gain Ratio

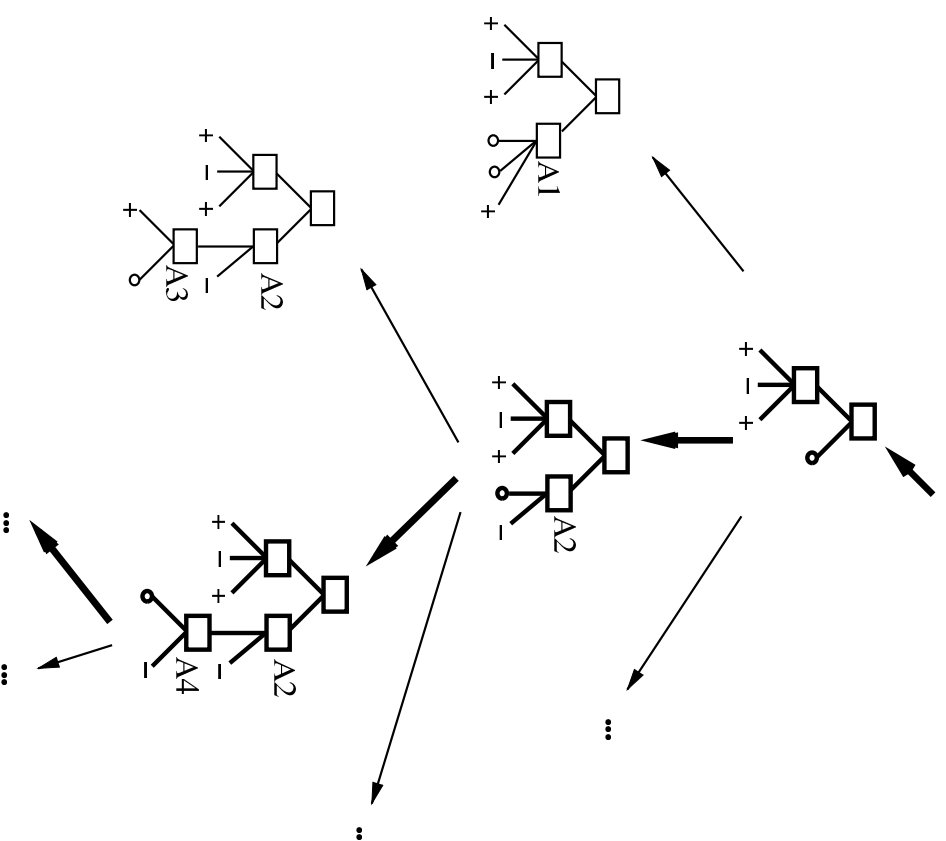
*GainRatio* non risolve tutti i problemi. Infatti potrebbe succedere che attributi significativi, ma che assumono qualche valore in più rispetto agli altri, potrebbero essere sfavoriti.

Di solito la strategia utilizzata è la seguente:

1. si calcola il *Guadagno* per ogni attributo;
2. si calcola la media dei guadagni calcolati;
3. si selezionano SOLO gli attributi che hanno *Guadagno* al di sopra della media;
4. si sceglie, fra gli attributi selezionati, quello che ha *GainRatio* maggiore.

# Apprendimento di Alberi di Decisione: Bias Induttivo

Il Bias Induttivo è sulla ricerca !



## Attributi Continui

Fino ad ora abbiamo considerato attributi a valori discreti.

**Cosa succede se un attributo è a valori continui ?**

**Esempio**

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	90 (Hot)	High	Weak	No
D2	Sunny	80 (Hot)	High	Strong	No
D3	Overcast	72 (Hot)	High	Weak	Yes
D4	Rain	60 (Mild)	High	Weak	Yes
D5	Rain	40 (Cool)	Normal	Weak	Yes
D6	Rain	48 (Cool)	Normal	Strong	No
D7	Overcast	40 (Cool)	Normal	Strong	Yes
D8	Sunny	60 (Mild)	High	Weak	Yes
D9	Sunny	40 (Cool)	Normal	Weak	Yes
...	...	...	...	...	...

## Attributi Continui

**Soluzione:** a partire dall'attributo  $A$ , creare dinamicamente l'attributo booleano

$$A_c = \begin{cases} true & \text{se } A < c \\ false & \text{altrimenti} \end{cases}$$

Come selezionare il valore "giusto" per  $c$  ?

Possibile soluzione: scegliere il  $c$  che corrisponde al *Guadagno* massimo !

E' stato dimostrato che il valore ottimo (massimo del *Guadagno*) si localizza nel valore di mezzo fra due valori con target diverso:

Esempio: tagli candidati

esempi	{D5,D7,D9}	{D6}	{D4,D8}	{D3}	{D2}	{D1}
<b>valore</b>	40	48	60	72	80	90
<b>target</b>	yes	no	yes	yes	no	no
<b>valore taglio</b>	↑ (44)	↑	↑	↑	↑	↑ (76)

## Attributi Continui

Quindi basta calcolare il *Guadagno* per

$$c = 44 \quad \underbrace{\{D5, D7, D9\}}_{T_{emp} < 44} \quad \underbrace{\{D1, D2, D3, D4, D6, D8\}}_{T_{emp} \geq 44} \quad \text{Guadagno} \simeq \boxed{0.379}$$

$$c = 54 \quad \underbrace{\{D5, D6, D7, D9\}}_{T_{emp} < 54} \quad \underbrace{\{D1, D2, D3, D4, D8\}}_{T_{emp} \geq 54} \quad \text{Guadagno} \simeq 0.091$$

$$c = 76 \quad \underbrace{\{D3, D4, D5, D6, D7, D9\}}_{T_{emp} < 76} \quad \underbrace{\{D1, D2, D8\}}_{T_{emp} \geq 76} \quad \text{Guadagno} \simeq 0.093$$

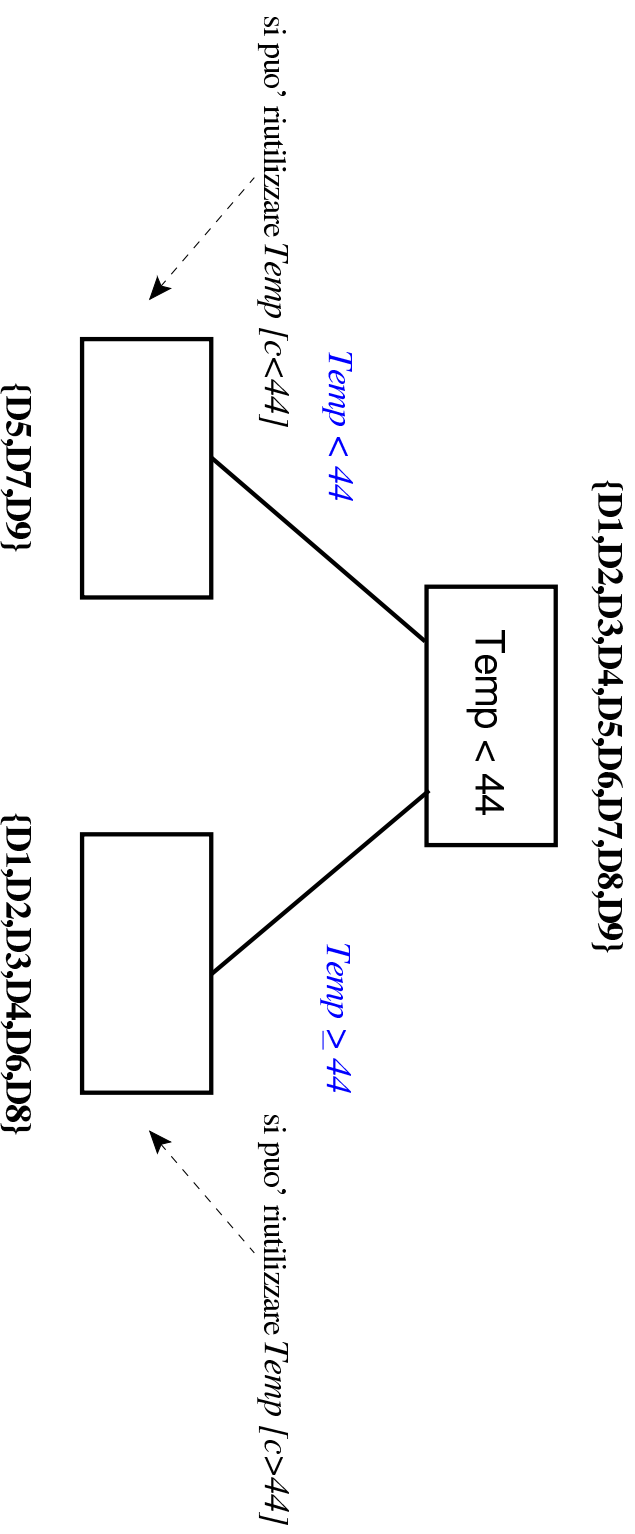
Quindi si seleziona  $c = 44$  poiché ottiene guadagno massimo !

Test finale:  $T_{emp} < 44$  ?

## Attributi Continui

**ATTENZIONE !**

Lo stesso attributo può essere riutilizzato sullo stesso cammino ...



Ovviamente, invece di utilizzare *Guadagno* si può anche utilizzare *GainRatio*



## Attributi con Costi

**Osservazione:** verificare il valore che un attributo assume può avere un costo !

Esempio: Diagnosi Medica

Attributo	Costo
esami sangue	€ 7
radiografi a	€ 15
visita medica	€ 40
T.A.C.	€ 150

E' possibile tener conto di tali costi nel costruire l'albero di decisione ?

Preferire attributi poco costosi: se possibile posizionarli in prossimità della radice (probabilità di verificarli più alta; es. probabilità 1 per la radice)

Usare un criterio di selezione dell'attributo (ottimo) che inglobi i costi

Esempio per la Diagnosi Medica:

$$\frac{2^{Guadagno(S,A)} - 1}{(Costo(A)+1)^w}$$

con  $w \in [0, 1]$

Esempio per la Percezione Robotica:

$$\frac{Guadagno^2(S,A)}{Costo(A)}$$

## Attributi con Valori Mancanti

**Problema:** nelle applicazioni pratiche può succedere che per alcuni esempi, alcuni attributi non abbiano alcun valore assegnato (valore mancante)

**Esempio: Diagnosi Medica**

- per il paziente 38 manca il risultato della T.A.C.;
- per il paziente 45 mancano gli esami del sangue e la radiografia;

Possibili soluzioni: dato un nodo  $n$  con associato l'insieme di esempi  $\hat{T}^r$ , quando per un esempio  $(x, target) \in \hat{T}^r$  manca il valore di  $A$

- utilizzare per  $A$  il valore in  $Valori(A)$  più frequente in  $\hat{T}^r$ ;
- come a), però considerare solo esempi con target uguale a quello dell'esempio corrente
- considerare tutti i valori  $a_i \in Valori(A)$  e la loro probabilità di occorrere  $prob(a_i | \hat{T}^r)$ , stimata su  $\hat{T}^r$ , e sostituire l'esempio  $(x, target)$  con  $|Valori(A)|$  istanze "frazionarie", una per ogni valore  $a_i \in Valori(A)$  (dove  $A = a_i$ ) e peso uguale a  $prob(a_i | \hat{T}^r)$

## Attributi con Valori Mancanti

Esempio a)

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	-	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes

Consideriamo l'attributo Outlook e l'esempio D5:

esempio originario

$D5 \equiv (-, Cool, Normal, Weak) \rightarrow D5' \equiv ($  Sunny  $, Cool, Normal, Weak)$

nuovo esempio

infatti:  $prob(Sunny|\hat{T}r) = 4/8$ ,  $prob(Overcast|\hat{T}r) = prob(Rain|\hat{T}r) = 2/8$

## Attributi con Valori Mancanti

Esempio b)

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	-	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes

Consideriamo l'attributo Outlook e l'esempio D5:

esempio originario

nuovo esempio

$$D5 \equiv (-, Cool, Normal, Weak) \rightarrow D5' \equiv ( \boxed{Overcast}, Cool, Normal, Weak)$$

infatti:

$$prob(Overcast|target = yes, \hat{T}r) = 2/4,$$

$$prob(Sunny|target = yes, \hat{T}r) = prob(Rain|target = yes, \hat{T}r) = 1/4$$

## Attributi con Valori Mancanti

Esempio c)

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	-	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes

Consideriamo l'attributo Outlook e l'esempio D5:

esempio originario

esempio frazionario

peso= $prob(a_i | \hat{T}^r)$

↗  $D5_S \equiv (Sunny, Cool, Normal, Weak)$       4/8

$D5 \equiv (-, Cool, Normal, Weak) \rightarrow$   $D5_O \equiv (Overcast, Cool, Normal, Weak)$       2/8

↘  $D5_R \equiv (Rain, Cool, Normal, Weak)$       2/8

## Attributi con Valori Mancanti

Esempio c)

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5'	-	Cool	Normal	-	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes

Consideriamo l'esempio D5' e gli attributi Outlook e Wind

Se pi`u attributi hanno valori mancanti, si continua a frazionare gli esempi gi`a frazionati ed il peso associato all'esempio frazionato è dato dal prodotto dei pesi ottenuti considerando un solo attributo a turno

## Attributi con Valori Mancanti

### Esempio c)

esempio originario

$D5' \equiv (-, Cool, Normal, -)$

esempio frazionario

↘  $D5'_{S,W} \equiv (Sunny, Cool, Normal, Weak)$

↘  $D5'_{S,S} \equiv (Sunny, Cool, Normal, Strong)$

↘  $D5'_{O,W} \equiv (Overcast, Cool, Normal, Weak)$

↘  $D5'_{O,S} \equiv (Overcast, Cool, Normal, Strong)$

↘  $D5'_{R,W} \equiv (Rain, Cool, Normal, Weak)$

↘  $D5'_{R,S} \equiv (Rain, Cool, Normal, Strong)$

$prob(a_i, b_j | \hat{T}r)$

4/8 · 5/8

4/8 · 3/8

2/8 · 5/8

2/8 · 3/8

2/8 · 5/8

2/8 · 3/8

dove  $a_i \in Valori(Outlook)$  e  $b_j \in Valori(Wind)$

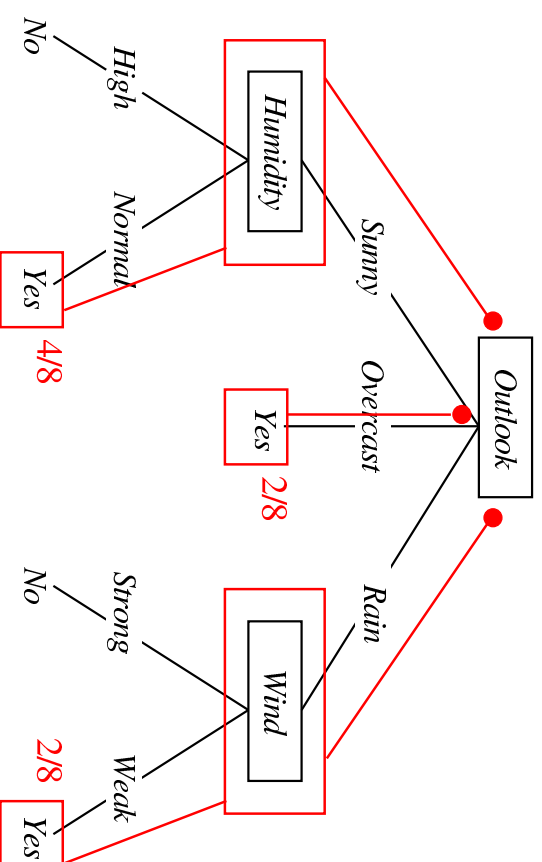
Si fa una assunzione di indipendenza degli attributi:

$$prob(a_i, b_j | \hat{T}r) = prob(a_i | \hat{T}r) \cdot prob(b_j | \hat{T}r)$$

## Attributi con Valori Mancanti

### Classificazione di D5 con metodo c)

Per ogni esempio frazionario si esegue la classificazione e per ogni possibile etichetta di classificazione si sommano i pesi degli esempi che raggiungono foglie con quella etichetta:



$$\text{Prob(Yes)} = 4/8 + 2/8 + 2/8 = 1$$

$$\text{Prob(No)} = 0$$



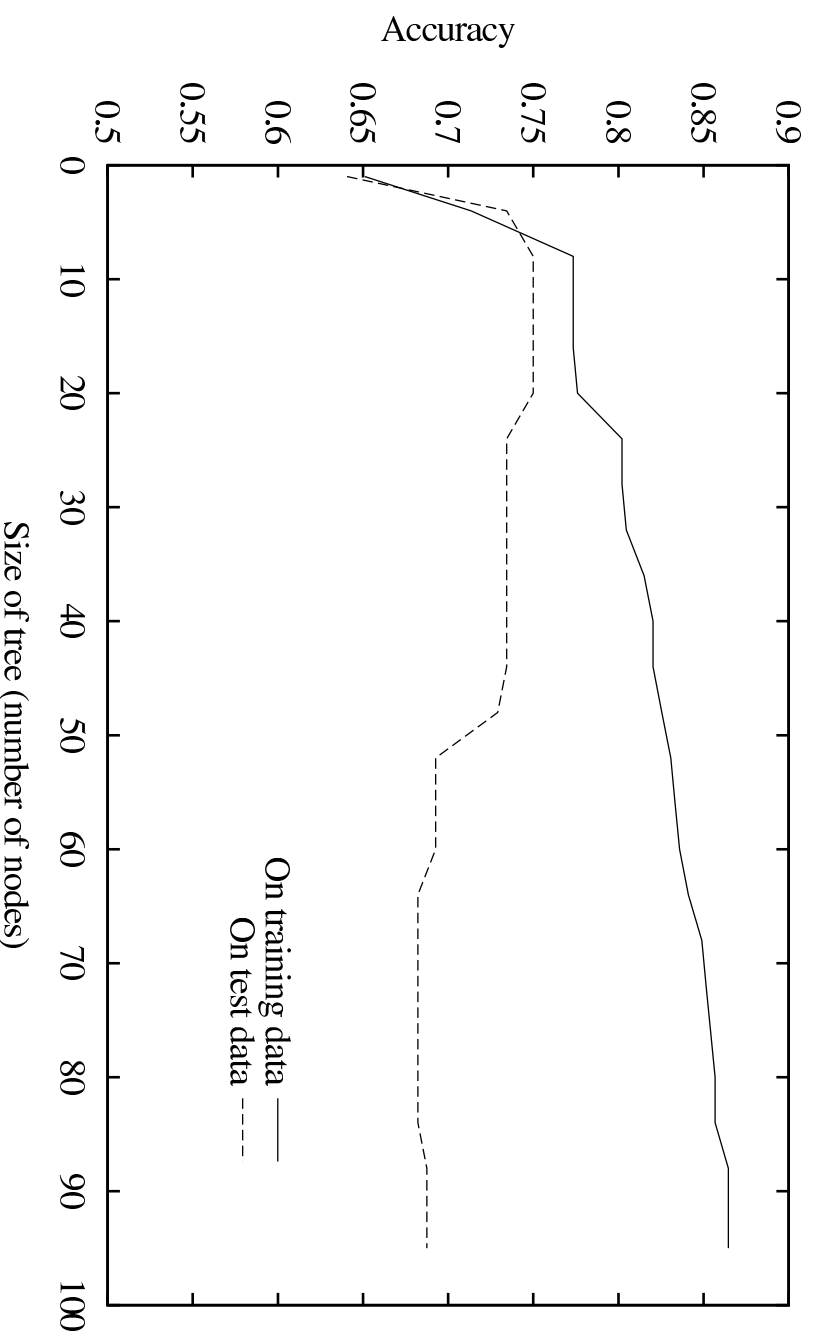
## Attributi con Valori Mancanti

### Apprendimento con metodo c)

- modificare il concetto di cardinalità di un insieme in modo da considerare i pesi frazionari;
- modificare la definizione di *Guadagno* conseguentemente;

## Apprendimento di Alberi di Decisione: Overfitting

**Problema:** Overfitting !



**Soluzione:** Algoritmi di Potatura (Pruning)...

## Potatura

**Problema 1:** Come effettuare la potatura ?

**Problema 2:** Quando fermarsi con la potatura (o alternativamente con l'apprendimento) ?

## Quando Fermarsi...

**Problema 1:** Come effettuare la potatura ?

**Problema 2:** Quando fermarsi con la potatura (o alternativamente con l'apprendimento) ?

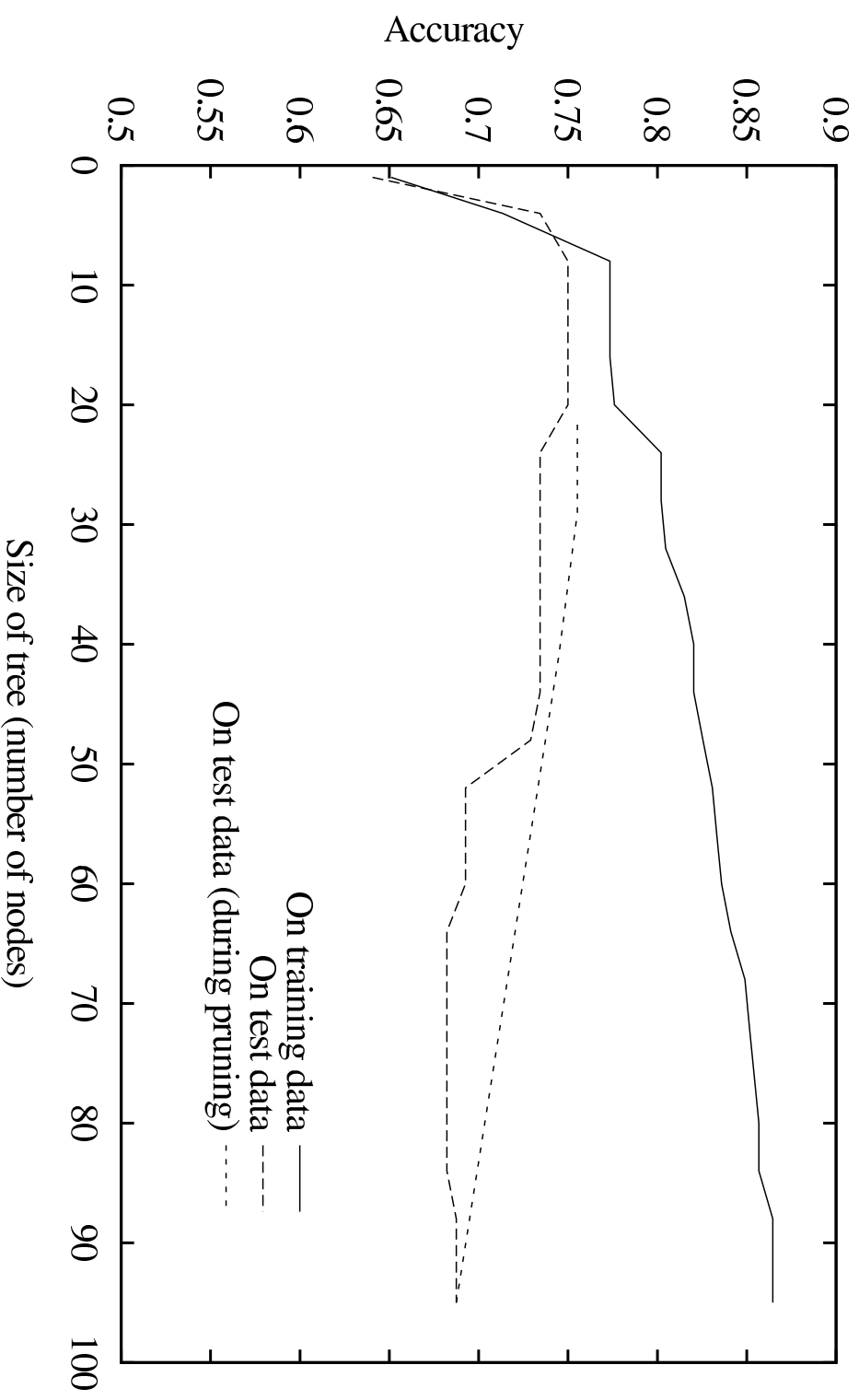
Consideriamo il **Problema 2**. Le principali “soluzioni” sono:

1. Valutare le prestazioni sull'insieme di apprendimento (usando un test statistico);
2. Valutare le prestazioni su un insieme (separato) di validazione;
3. Usare un principio di **minimizzazione della lunghezza di descrizione (MDL)**

$$\min_{Tree} [size(Tree) + size(errori(Tree))]$$

In ogni caso è difficile pervenire ad una soluzione ottima...

## Usando l'insieme di validazione



## Come Potare...

**Problema 1:** Come effettuare la potatura ? Le principali “soluzioni” sono:

### 1. **Reduced Error Pruning**

- Dividere  $T_r$  in  $T_{r1}$  e  $V_s$  (insieme di validazione);
  - Ripetere fin a quando le prestazioni peggiorano
  - (a) per ogni nodo (interno)  $n$  valutare l'impatto su  $V_s$  avendo potato il nodo (e i suoi discendenti);
  - (b) effettuare la potatura che porta alle prestazioni migliori su  $V_s$ ;
- (potatura: al sotto-albero radicato in  $n$  si sostituisce una foglia con etichetta uguale alla classe più frequente nell'insieme degli esempi associati al nodo  $n$ )

### 2. **Rule-Post Pruning**

## Rule-Post Pruning

L'idea di base è quella di trasformare un albero di decisione in un insieme di regole, e poi effettuare la potatura delle regole:

1. Si genera una regola  $R_i$  per ogni cammino *path*( $r, f_i$ ) dalla radice  $r$  alla foglia  $i$ -esima  $f_i$ ;  $R_i$  sarà nella forma

IF  $(Attr_{i_1} = v_{i_1}) \wedge (Attr_{i_2} = v_{i_2}) \wedge \dots \wedge (Attr_{i_k} = v_{i_k})$  THEN *label*  $f_i$

*precondizioni*

2. Si effettua la potatura indipendentemente su ogni regola  $R_i$ :
  - si stimano le prestazioni ottenute utilizzando SOLO  $R_i$  come classificatore;
  - si rimuovono le precondizioni (una o più) che conducono ad un aumento della stima delle prestazioni usando un approccio greedy;
3. Si ordinano le  $R_i$  potate per ordine decrescente di prestazione (evita conflitti); eventualmente, aggiunge come classificazione di default la classe più frequente;

## Classificare una nuova istanza

La **classificazione** di una nuova istanza da parte delle regole ordinate avviene seguendo l'ordine stabilito per le regole:

- la prima regola la cui precondizione è soddisfatta dalla istanza è usata per generare la classificazione
- se nessuna regola ha le condizioni soddisfatte, si utilizza la regola di default per classificare l'istanza (cioè si ritorna la classe più frequente nell'insieme di apprendimento);



## Considerazioni

Alcune considerazioni sul Rule-Post Pruning:

- la stima delle prestazioni necessaria per effettuare la potatura può essere fatta sia usando un insieme di validazione che utilizzando un test statistico sui dati di apprendimento;
- la trasformazione **Albero** → **Regole** permette di generare regole dove si possono considerare contesti per un nodo che non necessariamente contengono i suoi nodi avi (e in particolare la radice);
- di solito le regole sono più semplici da comprendere per un umano;

In genere il Rule-Post Pruning riesce a migliorare le prestazioni dell'albero di decisione di partenza e si comporta meglio del Reduced-Error Pruning