Apprendimento di Alberi di Decisione

Esempio di Algoritmo di Apprendimento: ID3

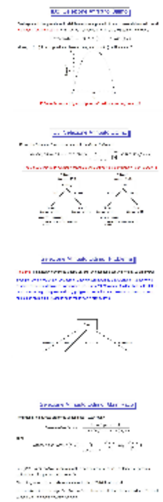
L'apprendimento di Alberi di Decisione tipicamente procede attraverso una procedura di tipo (divide et impera) che costruisce l'albero top-down:

1. crea il nodo radice, $\hat{Tr} \leftarrow Tr$ e inserisci tutti gli attributi nell'insieme $\mathcal{A}$;
2. se gli esempi in $\hat{Tr}$ sono tutti della stessa classe (- o +), assegna al nodo l'etichetta della classe e fermati;

   altrimenti

   (a) se $\mathcal{A}$ è vuoto, assegna al nodo l'etichetta della classe più frequente e fermati;

   altrimenti assegna al nodo $A \leftarrow$ $\boxed{\text{l'attributo “ottimo” in } \mathcal{A}}$;
3. partiziona $\hat{Tr}$ secondo i possibili valori che A può assumere: $\hat{Tr}_{A=val_1}, \ldots, \hat{Tr}_{A=val_{m(A)}}$, dove $m(A)$ — numero valori distinti di A;
4. $\forall \hat{Tr}_{A=val_j} = \emptyset$ crea una foglia figlio con l'etichetta della classe più frequente $\boxed{\text{in } \hat{Tr}}$;
5. $\forall \hat{Tr}_{A=val_i} \neq \emptyset$ crea nodo figlio e vai a 2 con.. $\boxed{\text{se necessario risalire ai nodi avi}}$



Criteri alternativi all'Entropia

Altri criteri suggeriti al posto dell'Entropia:

- **Variance Impurity** (per due classi)

$$p_- \cdot p_+$$

- **(Weighted) Gini Impurity** (generalizzazione di Variance Impurity a più di 2 classi)

$$\sum_{c,c' \in Classi, c \neq c'} \lambda_{c,c'} p_c \cdot p_{c'}$$

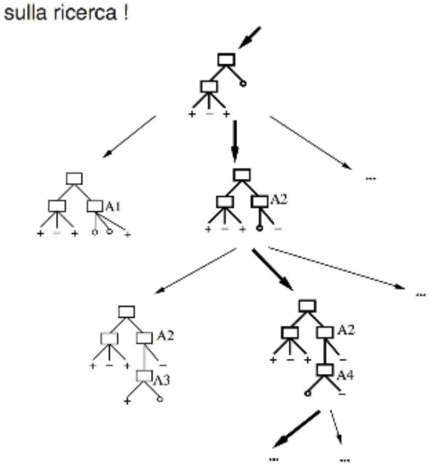
  dove $\lambda_{c,c'}$ sono parametri di costo per classificazioni errate

- **Misclassification Impurity**

$$1 - \max_{c \in Classi} p_c$$

Apprendimento di Alberi di Decisione: Bias Induttivo

Il Bias Induttivo è sulla ricerca !



Esempio di Algoritmo di Apprendimento: ID3

L'apprendimento di Alberi di Decisione tipicamente procede attraverso una procedura di tipo $\boxed{\text{(divide et impera)}}$ che costruisce l'albero top-down:

1. crea il nodo radice, $\hat{Tr} \leftarrow Tr$ e inserisci tutti gli attributi nell'insieme $\mathcal{A}$;
2. se gli esempi in $\hat{Tr}$ sono tutti della stessa classe (- o +), assegna al nodo l'etichetta della classe e fermati;

   altrimenti

   (a) se $\mathcal{A}$ è vuoto, assegna al nodo l'etichetta della classe più frequente e fermati;

   altrimenti assegna al nodo $A \leftarrow \boxed{\text{l'attributo "ottimo" in } \mathcal{A}}$;
3. partiziona $\hat{Tr}$ secondo i possibili valori che A può assumere:

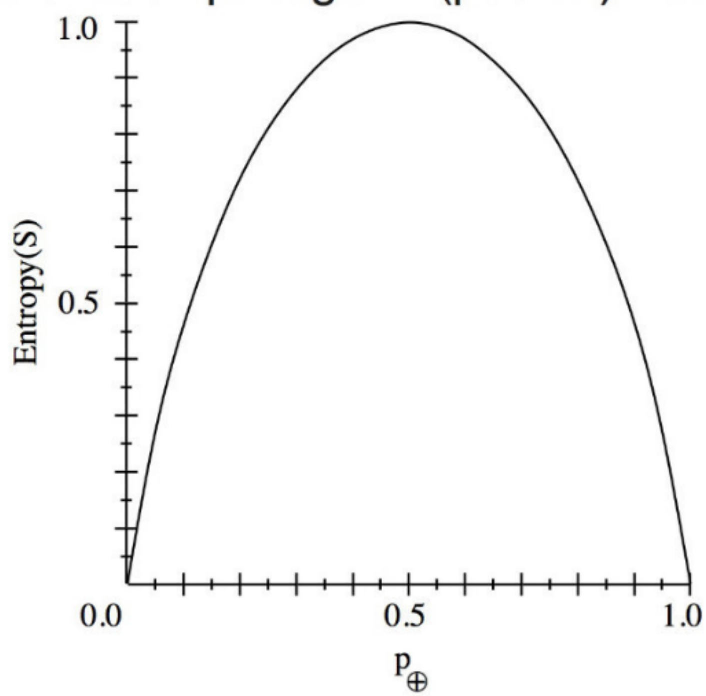
   $\hat{Tr}_{A=val_1}, \ldots, \hat{Tr}_{A=val_{m(A)}}$, dove $m(A) =$ numero valori distinti di A;
4. $\forall\, \hat{Tr}_{A=val_j} = \emptyset$ crea una foglia figlio con l'etichetta della classe più frequente $\boxed{\text{in } \hat{Tr}}$;
5. $\forall\, \hat{Tr}_{A=val_i} \neq \emptyset$ crea nodo figlio e vai a 2 con.. $\boxed{\text{se necessario risalire ai nodi avi}}$ $\Uparrow$

ID3: Selezione Attributo Ottimo

Vari algoritmi di apprendimento si differenziano soprattutto (ma non solo) dal modo in cui si seleziona l'attributo ottimo: ID3: utilizza il concetto di *Entropia* e *Guadagno Entropico*

$$Entropia(S) = -p_- log_2(p_-) - p_+ log_2(p_+)$$

dove p_- (p_+) è la proporzione di esempi negativi (positivi) nell'insieme S



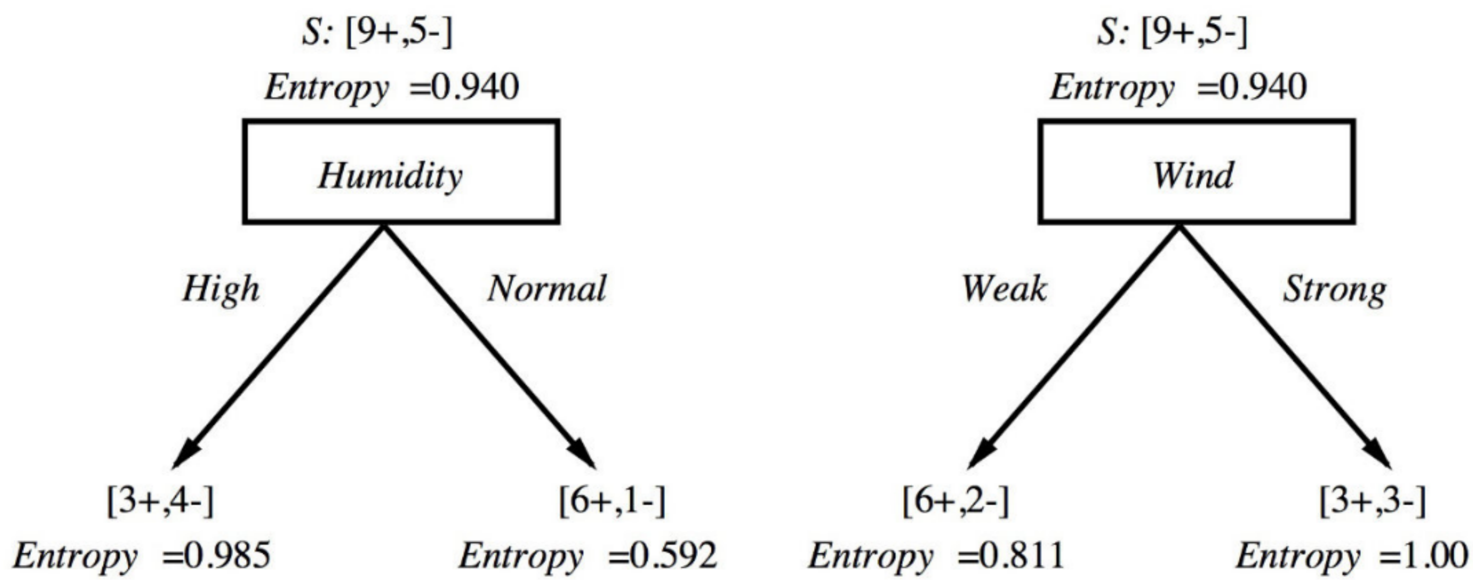
L' *Entropia* misura il "grado di purezza" dell'insieme degli esempi!

ID3: Selezione Attributo Ottimo

Si sceglie l'attributo A che massimizza il *Guadagno Entropico*:

$$Guadagno(S,A) = Entropia(S) - \sum_{v \in Valori(A)} \frac{|S_{A=v}|}{|S|} Entropia(S_{A=v})$$

Il *Guadagno* misura la riduzione aspettata della entropia nel partizionare i dati usando A



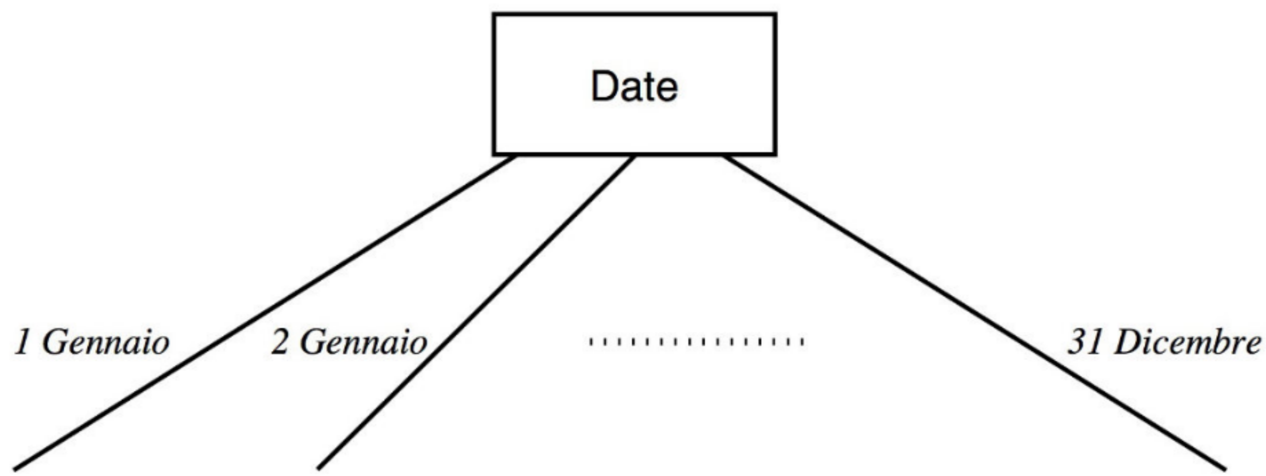
Gain (S, Humidity)
= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)
= .940 - (8/14).811 - (6/14)1.0
= .048

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*:

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

dove

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

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 $SplitInformation$ sfavorisce 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 attibuti 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.

Criteri alternativi all'Entropia

Altri criteri suggeriti al posto dell'Entropia:

- **Variance Impurity** (per due classi)

$$p_- \cdot p_+$$

- **(Weighted) Gini Impurity** (generalizzazione di Variance Impurity a più di 2 classi)

$$\sum_{c,c' \in Classi, c \neq c'} \lambda_{c,c'} p_c \cdot p_{c'}$$

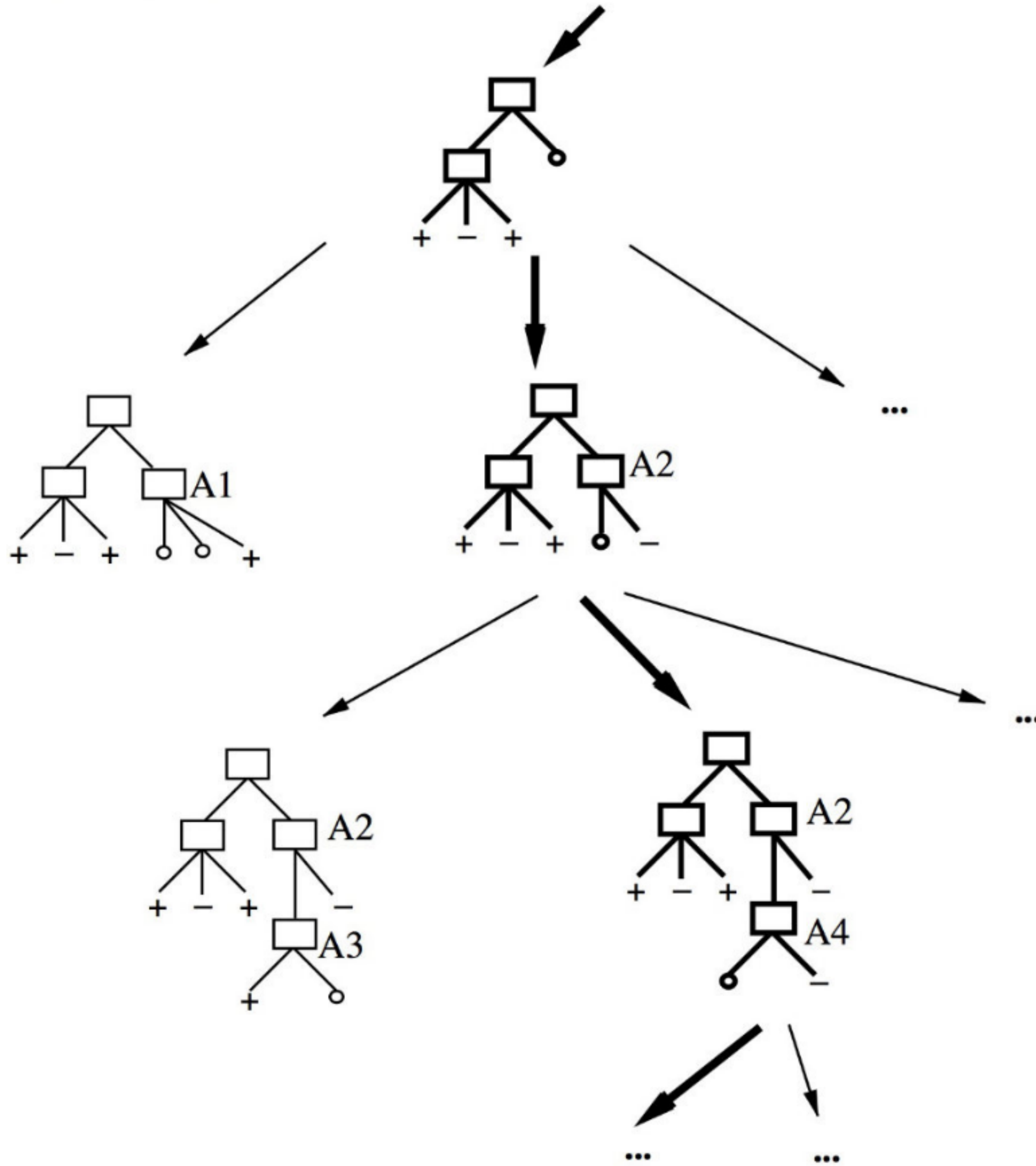
dove $\lambda_{c,c'}$ sono parametri di costo per classificazioni errate

- **Misclassification Impurity**

$$1 - \max_{c \in Classi} p_c$$

Apprendimento di Alberi di Decisione: Bias Induttivo

Il Bias Induttivo è sulla ricerca !



Casi Speciali

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 con Costi

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

Esempio: Diagnosi Medica

Attributo	Costo
esami sangue	€ 7
radiografia	€ 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: $\boxed{\frac{2^{Guadagno(S,A)}-1}{(Costo(A)+1)^w}}$ con $w \in [0,1]$

Esempio per la Percezione Robotica: $\boxed{\frac{Guadagno^2(S,A)}{Costo(A)}}$

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

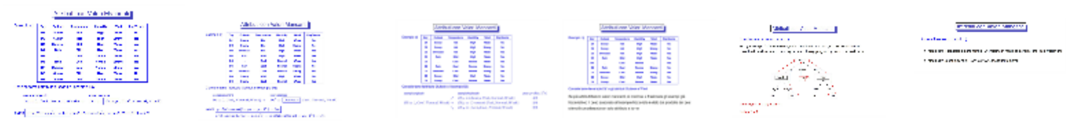
- per il paziente 38 manca il risultato della $\boxed{\text{T.A.C.}}$;
- per il paziente 45 mancano gli $\boxed{\text{esami del sangue}}$ e la $\boxed{\text{radiografia}}$;

Possibili soluzioni: dato un nodo n con associato l'insieme di esempi $\hat{Tr}$, quando per un esempio $(x, target) \in \hat{Tr}$ manca il valore di A

a) utilizzare per A il valore in $Valori(A)$ più frequente in $\hat{Tr}$;

b) come a), però considerare solo esempi con target uguale a quello dell'esempio corrente

c) considerare tutti i valori $a_i \in Valori(A)$ e la loro probabilità di occorrere $prob(a_i|\hat{Tr})$, stimata su $\hat{Tr}$, 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 $prob(a_i|\hat{Tr})$



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}
valore	40		48		60	72		80	90
target	yes		no		yes	yes		no	no
		↑		↑			↑		
valore taglio		(44)		(54)			(76)		

Attributi Continui

Quindi basta calcolare il $Guadagno$ per

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

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

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

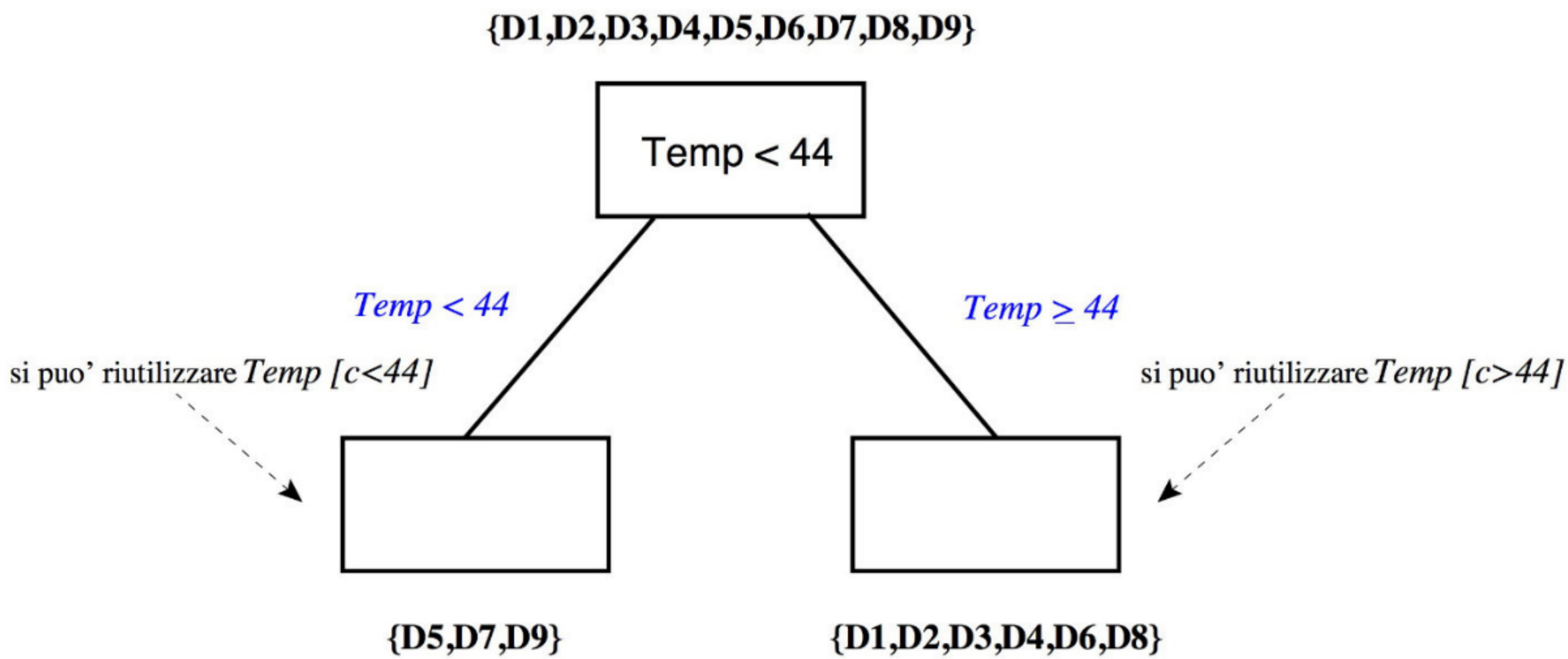
Quindi si seleziona $c = 44$ poiché ottiene guadagno massimo !

Test finale: $Temp < 44$?

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
radiografia	€ 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: $\boxed{\frac{2^{Guadagno(S,A)}-1}{(Costo(A)+1)^w}}$ con $w \in [0, 1]$

Esempio per la Percezione Robotica: $\boxed{\frac{Guadagno^2(S,A)}{Costo(A)}}$

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 $\hat{Tr}$, quando per un esempio $(x, target) \in \hat{Tr}$ manca il valore di A

a) utilizzare per A il valore in $Valori(A)$ più frequente in $\hat{Tr}$;

b) come a), però considerare solo esempi con target uguale a quello dell'esempio corrente

c) considerare tutti i valori $a_i \in Valori(A)$ e la loro probabilità di occorrere $prob(a_i|\hat{Tr})$, stimata su $\hat{Tr}$, 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 $prob(a_i|\hat{Tr})$

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 — nuovo esempio

$D5 \equiv (\_, Cool, Normal, Weak) \rightarrow \quad D5' \equiv (\boxed{Sunny}, Cool, Normal, Weak)$

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

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

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

infatti: $\boxed{prob(Overcast|target = yes, \hat{Tr}) = 2/4}$,

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

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=$prob(a_i \mid \hat{Tr})$
	$\nearrow$	$D5_S \equiv (Sunny, Cool, Normal, Weak)$	$4/8$
$D5 \equiv (\_, Cool, Normal, Weak)$	$\rightarrow$	$D5_O \equiv (Overcast, Cool, Normal, Weak)$	$2/8$
	$\searrow$	$D5_R \equiv (Rain, Cool, Normal, Weak)$	$2/8$

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

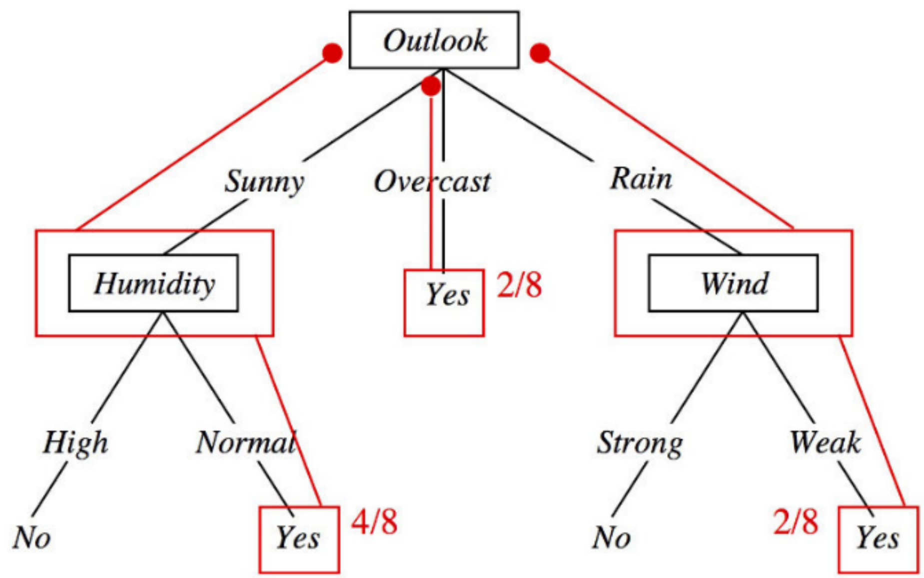
Consideriamo l'esempio D5' e gli attributi Outlook e Wind

Se più attributi hanno valori mancanti, si continua a frazionare gli esempi già 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

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:



Prob(Yes) = 4/8 + 2/8 + 2/8 = 1

Prob(No) = 0

Attributi con Valori Mancanti

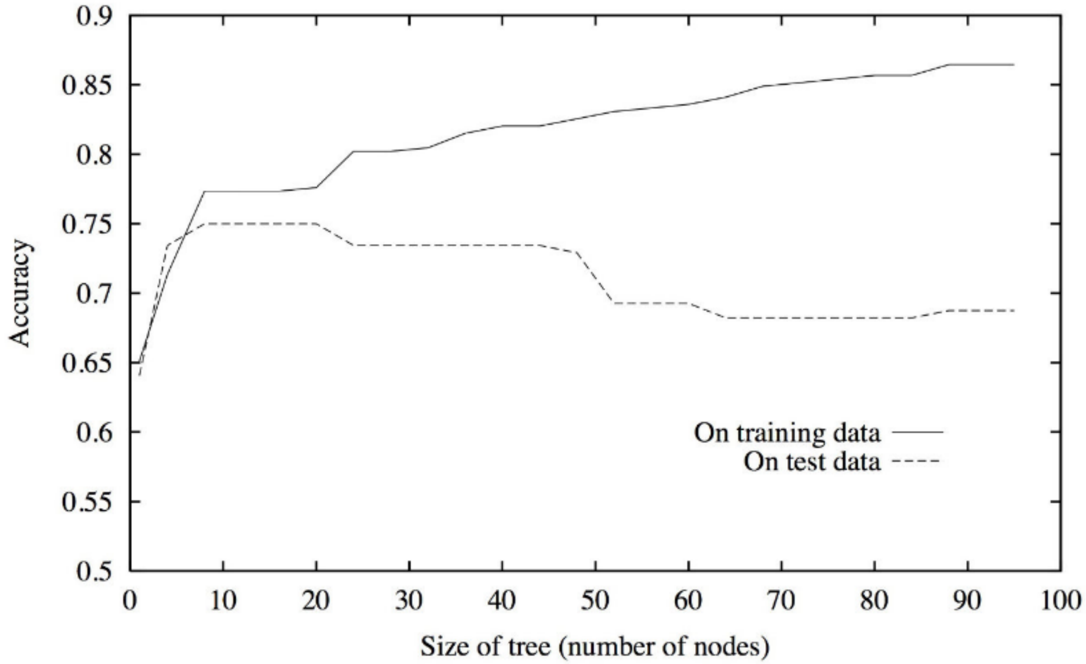
Apprendimento con metodo c)

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

Overfitting

Apprendimento di Alberi di Decisione: Overfitting

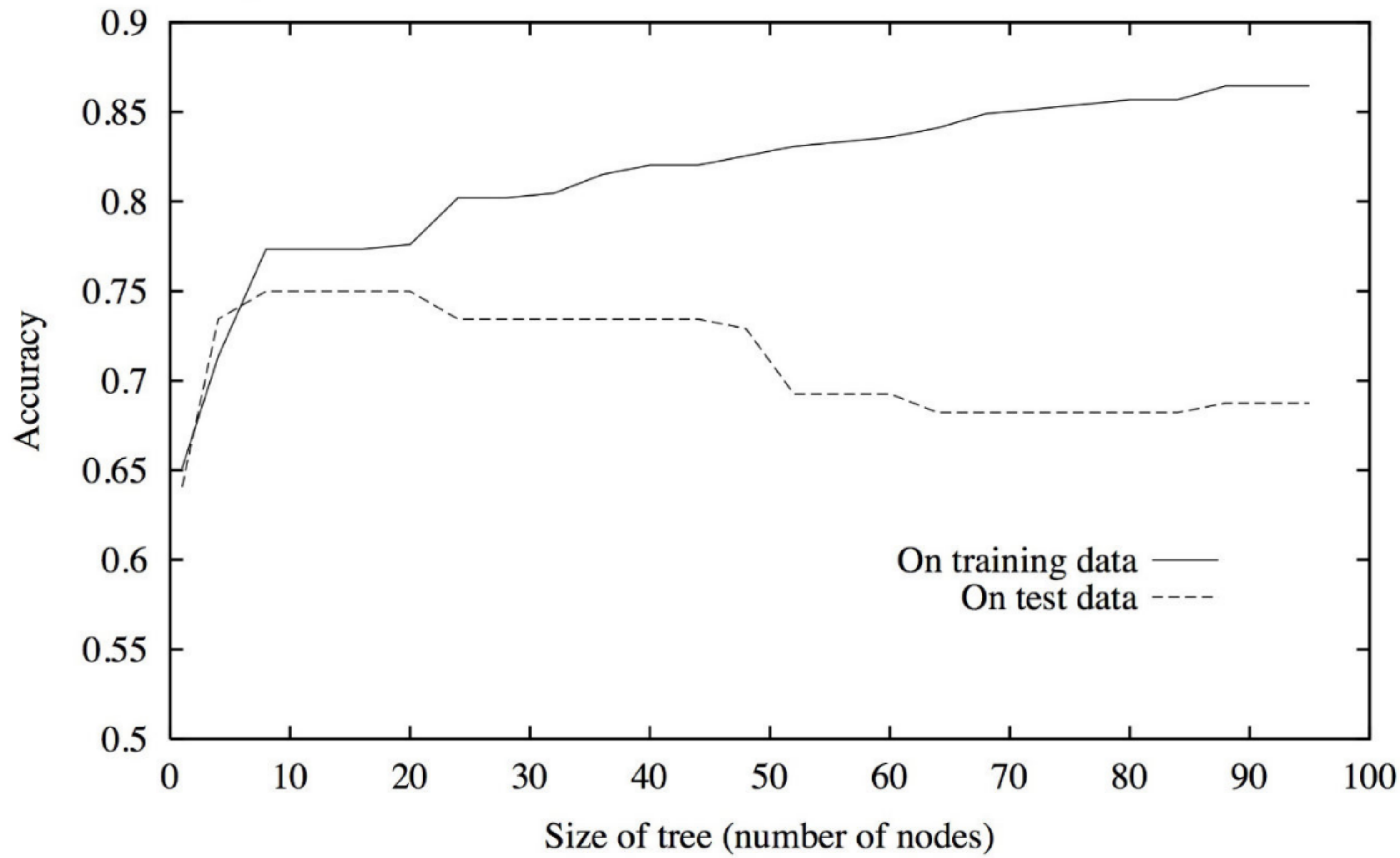
Problema: Overfitting !



Soluzione: Algoritmi di Potatura (Pruning)...

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) ?

Pruning

Quando Fermarsi...

Problema 1: Come effettuare la potatura ?

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

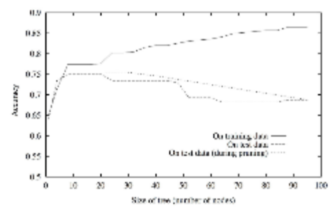
Consideriamo il **Problema 2**. Le principali "soluzioni" sono:

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)

$$\min_{Tree} \left[size(Tree) + size(errors(Tree)) \right]$$

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 Tr in Tr' e Va (insieme di validazione);
   - Ripetere fino a quando le prestazioni peggiorano

   (a) per ogni nodo (interno) n valutare l'impatto su Va avendo potato il nodo (e i suoi discendenti);

   (b) effettuare la potatura che porta alle prestazioni migliori su Va;

   (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. In alternativa si può usare la tecnica del "subtree raising")
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 $\underbrace{(Attr_{i_1} = v_{i_1}) \wedge (Attr_{i_2} = v_{i_2}) \wedge \cdots \wedge (Attr_{i_k} = v_{i_k})}_{precondizione}$ **THEN** $label_{f_i}$
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 (stima di accuratezza); 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

Quando Fermarsi...

Problema 1: Come effettuare la potatura ?

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

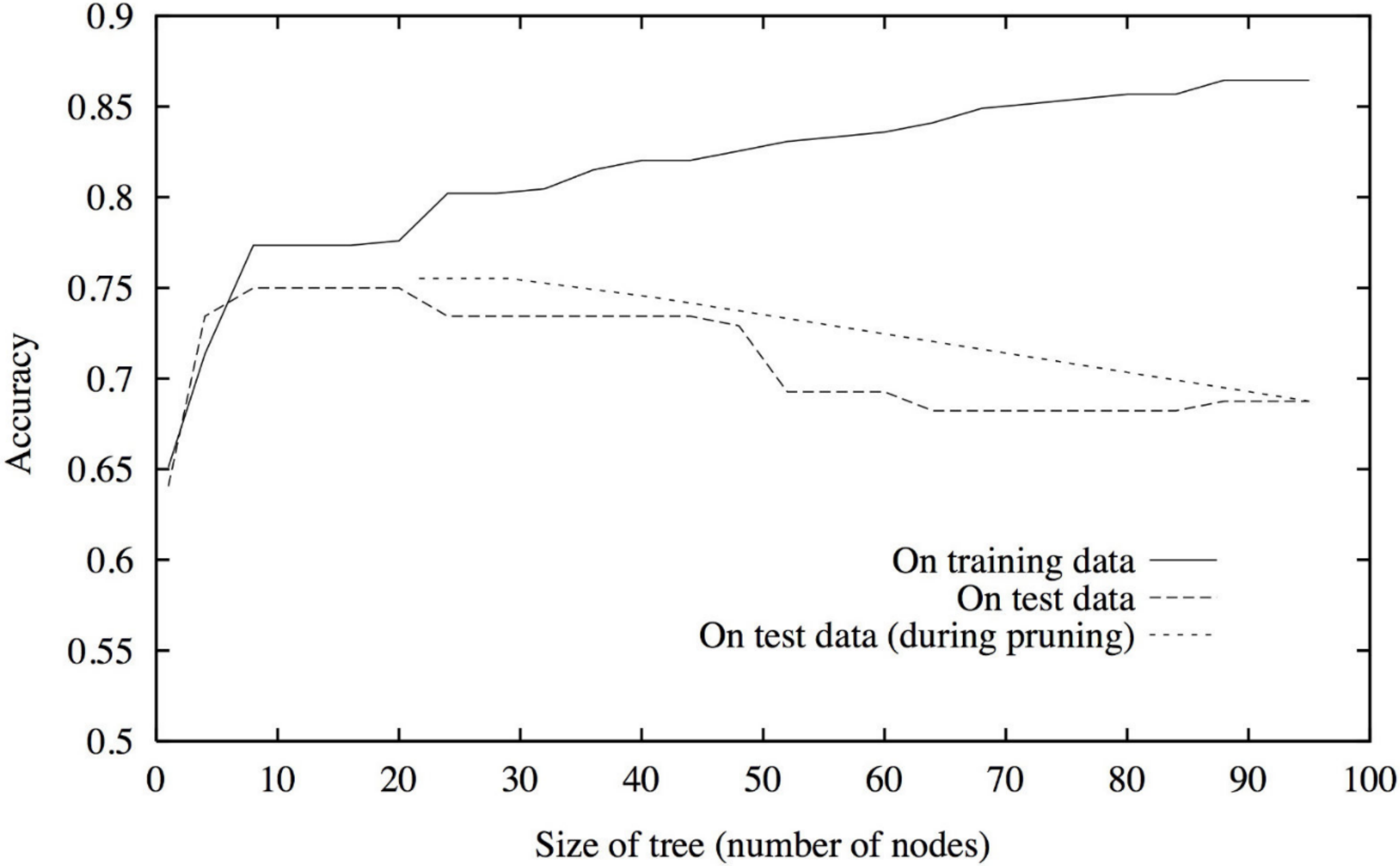
Consideriamo il Problema 2. Le principali "soluzioni" sono:

1. Valutare le prestazione 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)

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

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 Tr in $Tr\prime$ e Vs (insieme di validazione);
   - Ripetere fino a quando le prestazioni peggiorano

   (a) per ogni nodo (interno) n valutare l’impatto su Vs avendo potato il nodo (e i suoi discendenti);

   (b) effettuare la potatura che porta alle prestazioni migliori su Vs;

   (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. In alternativa si può usare la tecnica del “subtree raising”.)

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 $\underbrace{(Attr_{i_1} = v_{i_1}) \wedge (Attr_{i_2} = v_{i_2}) \wedge \cdots \wedge (Attr_{i_k} = v_{i_k})}_{precondizioni}$ THEN $label_{f_i}$

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 $\rightarrow$ 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