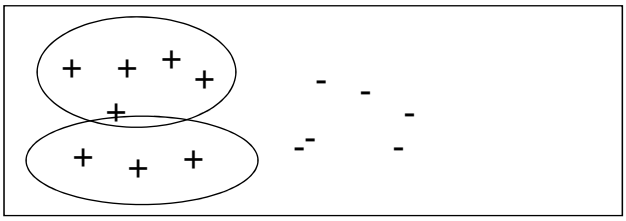
Apprendimento di insiemi disgiuntivi di regole

Apprendimento di insiemi disgiuntivi di regole

- Consideriamo prima l'apprendimento di un singolo concetto target (attributo target booleano).
- Regole della forma:
  - If Antecedente then Class=pos
- Classificazione: se nessuna regola fa match, allora classe negativa
- Metodo 1: Utilizzare un algoritmo di copertura sequenziale:
  - Ripeti finche' l'insieme di esempi positivi non e' vuoto
    - Apprendi una regola
    - Rimuovi gli esempi positivi coperti dalla regola
- Metodo 2: apprendere un albero di decisione e poi convertirlo in regole

Copertura Sequenziale

- Ad ogni esecuzione, si trova una regola congiuntiva che e' consistente con alcuni esempi positivi e con tutti gli esempi negativi
- Gli esempi positivi che sono stati considerati sono eliminati dal training set e l'algoritmo e' ripetuto finche' sono coperti tutti gli esempi positivi



Algoritmo di apprendimento sequenziale

APPRENDIMENTO_SEQUENZIALE(Attributo-target,Attributi,Esempi)

Regole_apprese←∅

Regola←APPRENDI_UNA_REGOLA(Attributo-target,Attributi,Esempi)

finche' l'insieme degli esempi positivi non e' vuoto fai

  Regole_apprese← Regole_apprese∪{Regola}

  Esempi ←Esempi-{esempi positivi coperti da Regola}

  Regola←APPRENDI_UNA_REGOLA(Attributo-target,Attributi,Esempi)

restituisci le Regole_apprese

APPRENDI_UNA_REGOLA

- APPRENDI_UNA_REGOLA accetta un insieme di esempi positivi e negativi come input, e produce una regola singola che copre molti esempi positivi e pochi o nessun esempio negativo
- Alta accuratezza, ma non necessariamente alta copertura
- Come implementare questa procedura?
- Una possibilita': per mezzo dell'algoritmo Candidate-Elimination

APPRENDI_UNA_REGOLA

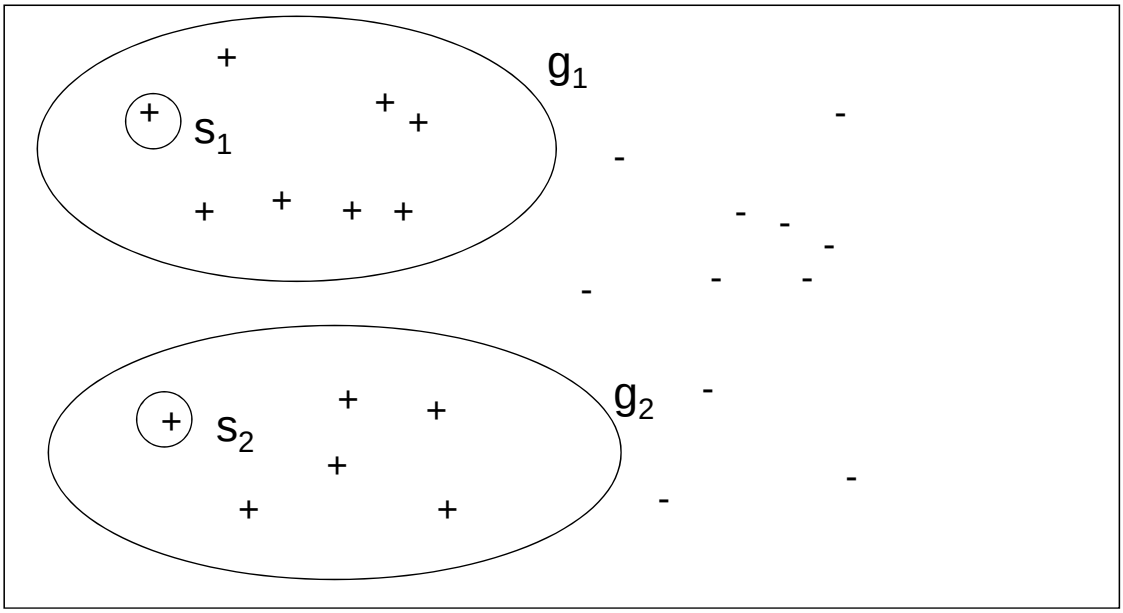
1-Inizializza S a contenere un esempio positivo (esempio seme)

2-Per ciascun esempio negativo applica la routine Update-G a G

3-Restituisci come regola una descrizione g da G

Dato che Update-G e' stato applicato usando tutti gli esempi negativi, g non copre nessun esempio negativo, ma puo' coprire diversi esempi positivi

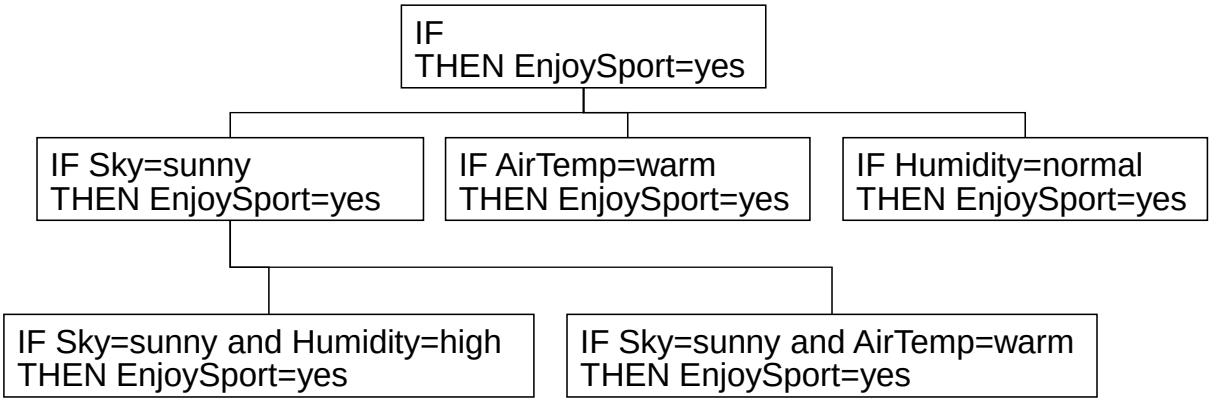
Copertura sequenziale + Candidate Elimination



Ricerca da generale a specifico

- Un approccio differente per realizzare APPRENDI_UNA_REGOLA consiste nel compiere una ricerca nello spazio delle regole partendo dalla regola più generale (con antecedente vuoto) e aggiungendo via via letterali all'antecedente
- Ad ogni passo si aggiunge il letterale che produce la regola più promettente
- Ricerca greedy top-down o da generale a specifico attraverso lo spazio delle regole

Spazio di ricerca



- Ad ogni passo viene scelta il letterale che migliora maggiormente la performance della regola.

Misure di performance

- In APPRENDI_UNA_REGOLA si puo' usare
  - Accuratezza sul training set: sia n il numero di esempi coperti dalla regola c e n_c il numero degli esempi classificati correttamente (ovvero il numero di e+ coperti dalla regola). Uguale alla precision della regola

$$P(+|c)=n_c/n$$

Questa stima prende il nome di stima attraverso la frequenza relativa

Misure di performance

- Problemi nel caso di pochi esempi:
  - se n_c=0 P(+|c)=0 anche se n e' molto piccolo
  - se n=0 P(+|c) diventa 0/0 e non si puo' calcolare
- Cestnik ha mostrato che in questi casi la stima attraverso la frequenza relativa non e' affidabile ed e' meglio usare la stima di Laplace: nel caso di due classi

$$P(+\mid c) = \frac{n_c + 1}{n + 2}$$

  - nel caso di N classi

$$P(+\mid c) = \frac{n_c + 1}{n + N}$$

Misure di performance

- La stima di Laplace assume che le 2 (o N) classi siano uniformemente distribuite nel training set.
- Se questo non e' vero bisogna usare la stima-m: sia p la probabilita' che un esempio estratto a caso dal training set abbia la classificazione assegnata dalla regole e m un peso

$$P(+\mid c) = \frac{n_c + mp}{n + m}$$

  - Se m=0 si ottiene l'accuratezza
  - Se m=N, p=1/N si ottiene la stima di Laplace

Misure di performance

- Per grandi valori di n_c ed n la stima di Laplace e la stima-m si avvicinano alla frequenza relativa

Ricerca beam

- La ricerca da generale a specifico e' generalmente una ricerca hill-climbing senza backtracking
- Pericolo di una scelta subottimale ad ogni passo
- Per ridurre il rischio viene a volte effettuata una ricerca beam
- Ad ogni passo, viene mantenuta una lista dei migliori k candidati, invece del singolo miglior candidato
- Sia CN2 [Clark&Niblett, 1989] che AQ [Michalski, 1969, 1986] compiono una ricerca beam top-down. AQ e' basato su un esempio seme per guidare la ricerca di regole.

Apprendimento di piu' concetti

- Caso in cui l'attributo target non e' booleano
- Non si apprendono piu' solamente regole che hanno nel conseguente Vero, ma si apprende un insieme di regole con diversi conseguenti
- Due approcci:
  - Lista ordinata di regole: la prima regola che fa match con l'esempio da classificare fornisce la classe. L'ultima regola e' una regola di default (CN2)
  - Lista non ordinata di regole: se l'esempio fa match con una sola regola, la classificazione e' data da quella regola. Se fa match con piu' regole, la classificazione e' data dalla piu' comune tra gli esempi di training coperti da quelle regole. Se non fa match con nessuna, la classificazione e' data dalla classe piu' frequente tra gli esempi di training (AQ)

Algoritmo di apprendimento sequenziale di CN2

- Si differenzia dall'algoritmo per l'apprendimento sequenziale visto prima perche':
  - E' in grado di imparare regole per diversi concetti
  - La procedura APPRENDI_UNA_REGOLA non restituisce una regola, ma solo un antecedente, il conseguente e' aggiunto successivamente

Algoritmo di apprendimento sequenziale di CN2

CN2(Attributo_target,Attributi,Esempi)

Regole_apprese←∅

Ripeti

Best_ant ←APPRENDI_UN_ANTECEDENTE (Attributo_target, Attributi, Esempi,k)

Sia E' l'insieme di esempi coperti da Best_ant

Esempi ←Esempi-E'

Sia C la classe piu' comune tra gli esempi di E'

Aggiungi la regola 'if Best_ant then class=C' alla fine di Regole_apprese

Finche' Best_ant e' nullo o Esempi e' vuoto

Aggiungi alla fine di Regole_apprese la regola di default 'class=C' dove C e' la classe piu' frequente in Esempi

restituisci le Regole_apprese

APPRENDI_UN_ANTECEDENTE (1)

APPRENDI_UN_ANTECEDENTE(Attributo_target,Attributi, Esempi,k)

Beam←{∅}

Best_ant ← ∅

Finche' Beam non e' vuoto fai

Specializza tutti gli antecedenti di Beam:

All_constraints ←insieme di tutti i vincoli della forma a=v

Nuovo_beam ←per ciascun h in Beam, per ciascun c in All_constraints: crea una specializzazione di h aggiungendo c

Rimuovi da Nuovo_beam tutte le ipotesi che sono duplicate, inconsistenti o che erano gia' in Beam

APPRENDI_UN_ANTECEDENTE (2)

Aggiorna Best_ant:

Per ciasun antecedente h in Nuovo_beam

se Performance(h,Esempi,Attributo_target)> Performance(Best_ant,Esempi,Attributo_target) then Best_ant ←h

Aggiorna Beam:

Beam ← i migliori k membri di Nuovo_beam secondo la misura di Performance.

Fine ciclo

Ritorna Best_ant

CN2

- Dato un concetto c, gli esempi positivi per gli altri concetti sono esempi negativi per c
- In questo caso, togliendo gli esempi positivi per un concetto c nel ciclo di covering si rimuovono anche degli esempi negativi per gli altri concetti
- Rischio di generare regole troppo generali per gli altri concetti
- Questo problema e' risolto dal fatto che le regole sono valutate in ordine. Quindi, anche se le ultime regole sono molto generali questo non e' un problema perche' le regole precedenti intercettano gli esempi.

Misure di performance

- In APPRENDI_UN_ANTECEDENTE si usa
  - Entropia

$$info(S) = -\sum_{j=1}^{k} \frac{freq(c_j, S)}{|S|} \times \log_2 \left(\frac{freq(c_j, S)}{|S|} \right)$$

- Dove S e' il training set e c_j sono le k classi
- Di solito si considera –info(S) in modo che le clausole migliori abbiamo i valori piu' alti

Esempio

No	Outlook	Temp	Humid	Windy	Class
D1	sunny	mild	normal	T	P
D2	sunny	hot	high	T	N
D3	sunny	hot	high	F	N
D4	sunny	mild	high	F	N
D5	sunny	cool	normal	F	P
D6	overcast	mild	high	T	P
D7	overcast	hot	high	F	P
D8	overcast	cool	normal	T	P
D9	overcast	hot	normal	F	P
D10	rain	mild	high	T	N
D11	rain	cool	normal	T	N
D12	rain	mild	normal	F	P
D13	rain	cool	normal	F	P
D14	rain	mild	high	F	P

Esempio

- Apprendi un antecedente:

Beam←{∅}

Best_ant ← ∅

Specializza gli antecedenti di Beam:

Nuovo_beam ←{outlook=sunny, outlook=overcast, outlook=rain, temp=hot, temp=mild, temp=cool, humid=high, humidity=normal, windy=t, windy=f}

Esempio

Calcola la performance di ciascun antecedente

outlook=sunny	-info(S)=$2/5\log_2 2/5+3/5\log_2 3/5$=-0.97
outlook=overcast	-info(S)=0
outlook=rain	-info(S)=-0.97
temp=hot	-info(S)=-1
temp=mild	-info(S)=-0.92
temp=cool	-info(S)=-0.81
humid=high	-info(S)=-0.99
humid=normal	-info(S)=-0.59
wind=t,	-info(S)=-1
wind=f	-info(S)=-0.81

Esempio

- Best_ant ← outlook=overcast
- Aggiorna Beam (dimensione=2):
- Beam ← {outlook=overcast, humid=normal }
- Nuova iterazione:
- Nuovo_beam ←{

outlook=overcast and temp=hot,
outlook=overcast and temp=mild,
outlook=overcast and temp=cool,
outlook=overcast and humid=high,
outlook=overcast and humid=normal,
outlook=overcast and windy=t,
outlook=overcast and windy=f,

Esempio

humid=normal and outlook=sunny,
humid=normal and outlook=overcast,
humid=normal and outlook=rain,
humid=normal and temp=hot,
humid=normal and temp=mild,
humid=normal and temp=cool,
humid=normal and windy=t,
humid=normal and windy=f}

Esempio

- Calcola la performance di ciascun antecedente

outlook=overcast and temp=hot, -info(S)=0
outlook=overcast and temp=mild, -info(S)=0
outlook=overcast and temp=cool, -info(S)=0
outlook=overcast and humid=high, -info(S)=0
outlook=overcast and humid=normal, -info(S)=0
outlook=overcast and windy=t, -info(S)=0
outlook=overcast and windy=f, -info(S)=0

Esempio

humid=normal and outlook=sunny, -info(S)=0
humid=normal and outlook=overcast, -info(S)=0
humid=normal and outlook=rain, -info(S)=-0.92
humid=normal and temp=hot, -info(S)=0
humid=normal and temp=mild, -info(S)=0
humid=normal and temp=cool, -info(S)=-0.81
humid=normal and windy=t, -info(S)=-0.92
humid=normal and windy=f -info(S)=0

Esempio

- Best_ant rimane outlook=overcast
- Aggiorna Beam (dimensione=2):
- Beam ← {outlook=overcast and temp=hot,

outlook=overcast and temp=mild }

Esempio

- Nuova iterazione:
- Nuovo_beam ←{

outlook=overcast and temp=hot and humid=normal,
outlook=overcast and temp=hot and humid=high,
outlook=overcast and temp=hot and windy=t,
outlook=overcast and temp=hot and windy=f,
outlook=overcast and temp=mild and humid=normal,
outlook=overcast and temp=mild and humid=high,
outlook=overcast and temp=mild and windy=t,
outlook=overcast and temp=mild and windy=f}

Esempio

- Calcola la performance di ciascun antecedente

outlook=overcast and temp=hot and humid=normal 0
outlook=overcast and temp=hot and humid=high 0
outlook=overcast and temp=hot and windy=t 0
outlook=overcast and temp=hot and windy=f 0
outlook=overcast and temp=mild and humid=normal 0
outlook=overcast and temp=mild and humid=high 0
outlook=overcast and temp=mild and windy=t 0
outlook=overcast and temp=mild and windy=f 0

Esempio

- Best_ant rimane outlook=overcast
- Aggiorna Beam (dimensione=2):
- Beam ← {

outlook=overcast and temp=hot and humid=normal,
outlook=overcast and temp=hot and humid=high}

- Nuova iterazione:
- Nuovo_beam ←{

outlook=overcast and temp=hot and humid=normal and windy=t,
outlook=overcast and temp=hot and humid=normal and windy=f,
outlook=overcast and temp=hot and humid=high and windy=t,
outlook=overcast and temp=hot and humid=high and windy=t}

Esempio

- Calcola la performance di ciascun antecedente

outlook=overcast and temp=hot and humid=normal and windy=t 0
outlook=overcast and temp=hot and humid=normal and windy=f 0
outlook=overcast and temp=hot and humid=high and windy=t 0
outlook=overcast and temp=hot and humid=high and windy=t 0

- Best_ant rimane outlook=overcast
- Aggiorna Beam (dimensione=2):
- Beam ← {

outlook=overcast and temp=hot and humid=normal and windy=t,
outlook=overcast and temp=hot and humid=normal and windy=f
}

Esempio

- Nuova iterazione:
- Nuovo_beam ← ∅
- Fine del ciclo
- Restituisci Best_ant=outlook=overcast
- Nel ciclo esterno:
  - Gli es coperti da outlook=overcast sono D6, D7, D8 e D9
  - Rimuovi da E questi esempi
  - Aggiungi la regola 'if outlook=overcast then class=P' alla teoria
  - Esegui un'altra iterazione del ciclo esterno
  - .....

Apprendimento di piu' concetti da parte di AQ

- Si apprende un concetto alla volta usando come esempi negativi gli esempi positivi per gli altri concetti usando l'algoritmo APPRENDIMENTO_SEQUENZIALE
  - Dopo aver appreso le regole per un concetto, l'insieme di esempi per il concetto viene riportato allo stato iniziale (tutti gli esempi)
- L'unione degli insiemi di regole appresi per ciascun concetto forma l'output del sistema di apprendimento

Determinazione delle regole di produzione da un albero decisionale

- Da un albero decisionale puo' essere generato un insieme di regole, una per ogni foglia, componendo tutti i test dalla radice dell'albero alle foglie
- Possibili successivi passi di semplificazione

Esempio

- Dall'albero

Outlook=sunny
    | humidity ≤ 75: P
    | humidity > 75: N
Outlook=overcast: P
Outlook=rain
    | windy=True: N
    | windy=False: P

- Si estraggono le regole

If Outlook=sunny and humidity ≤ 75 then Class=P

If Outlook=sunny and humidity > 75 then Class=N

if Outlook=overcast then Class=P

if Outlook=rain and windy=True then Class=N

if Outlook=rain and windy=False then Class=P

Copertura sequenziale simultanea vs sequenziale

Gli algoritmi di apprendimento di alberi di decisione possono essere considerati algoritmi di copertura simultanea, in contrasto con gli algoritmi di copertura sequenziale.

- Gli algoritmi di copertura sequenziale sono più lenti.
- Per imparare un insieme di n regole, ciascuna contenente k test attributo-valore nelle loro precondizioni, gli algoritmi di copertura sequenziale devono compiere n*k passi di ricerca primitivi (scelte indipendenti)

Copertura sequenziale simultanea vs sequenziale

- Gli algoritmi di copertura simultanea sono generalmente piu' veloci, dato che ogni test con m possibili risultati contribuisce alla scelta delle precondizioni per almeno m regole
- Se ogni test ha m risultati e ho n regole, la profondita' dell'albero sara' $\log_m(n)$ e le regole hanno ciascuna $\log_m(n)$ test. Il numero di test da scegliere sara' quindi:

$$\sum_{i=0}^{\log_m n-1} m^i = \frac{m^{\log_m(n)}-1}{m-1} = \frac{n-1}{m-1} \xrightarrow[n\to\infty]{} \frac{n}{m-1}$$

- Con la copertura sequenziale: n* $\log_m(n)$ scelte

Copertura sequenziale simultanea vs sequenziale

- Gli algoritmi di copertura sequenziale compiono un grande numero di scelte indipendenti, mentre gli algoritmi di copertura simultanea compiono un numero ristretto di scelte dipendenti.

Apprendere regole direttamente o convertire un albero in regole?

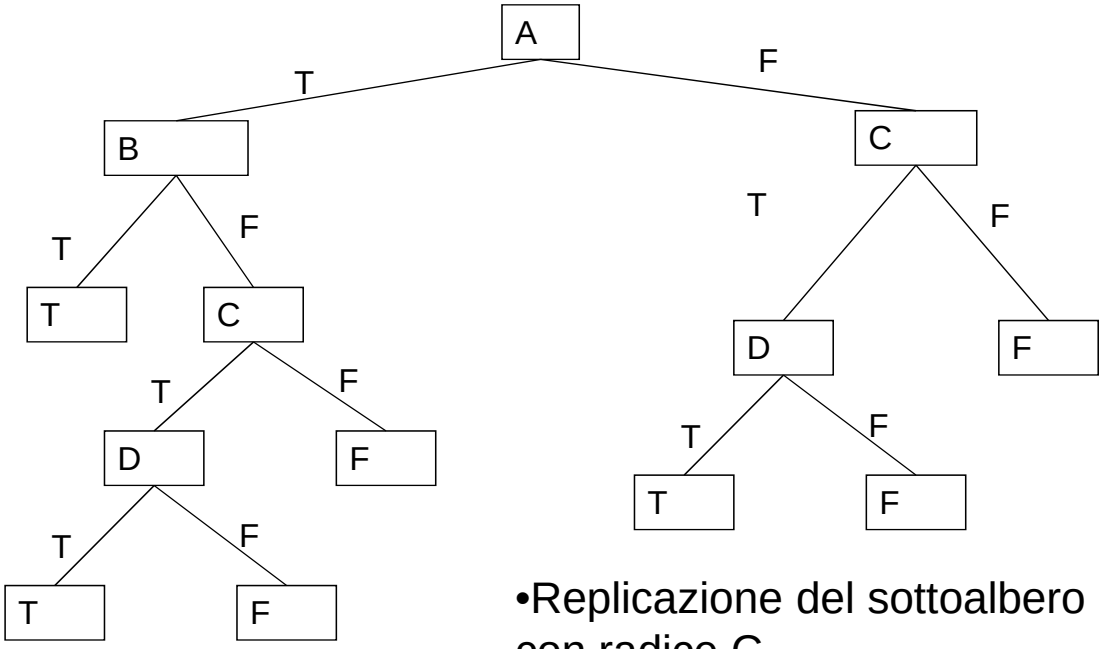
- A seconda della quantità di dati a disposizione:
  - Se si hanno molti dati, si possono usare gli algoritmi di copertura sequenziale perche' i dati possono supportare le molteplici scelte indipendenti
  - Se i dati sono scarsi, sono meglio gli algoritmi di copertura simultanea in quanto vengono usati tutti i dati nelle scelte

Apprendere regole direttamente o convertire un albero in regole?

- A seconda del tipo di concetto da apprendere
  - Se il concetto è altamente disgiuntivo con molte condizioni indipendenti, l'apprendimento di alberi ha basse prestazioni nel caso di dati scarsi

Concetti disgiuntivi rappresentati con alberi di decisione

- Albero che rappresenta il concetto AB ∨ CD



•Replicazione del sottoalbero con radice C

Concetti digiuntivi rappresentati mediante regole

- Concetto AB ∨ CD:
  - If A=T and B=T then class=T
  - If C=T and D=T then class=T
  - class=F

Apprendimento di regole con c4.5

- c4.5 e' in grado di apprendere anche regole (con il programma c4.5rules)
- Non converte semplicemente un albero in regole ma effettua una serie di elaborazioni
  - Generalizzazione delle regole singole
  - Selezione delle regole
  - Ordinamento delle regole
  - Scelta della regola di default
  - Seconda selezione di regole
- L'obiettivo di queste elaborazioni e' quello di ottenere un classificatore che sia piu' comprensibile per gli esseri umani dell'albero e che sia altrettanto accurato sui casi non visti

Generalizzazione delle regole

- Ad esempio la piu' profonda foglia T dell'albero precedente corrisponde alla regola
  - if A=T and B=F and C=T and D=T then class=T
- L'antecedente delle regole puo' contenere condizioni irrilevanti
  - Ad esempio, rimuovendo le condizioni A=T, B=F si ottiene una regola con la stessa accuratezza (1)

Generalizzazione delle regole

- Come decidere se cancellare una condizione nell'antecedente?
- Sia R un regola della forma

  If A then class C
- e R^- una regola piu' generale della forma

  If A^- then class C
- dove A^- e' ottenuto da A cancellando una condizione X
- Vogliamo decidere se tenere X sulla base dei dati di training

Generalizzazione delle regole

- Ogni esempio che soddisfa A^- puo' o meno appartenere a C e puo' o meno soddisfare X
- Abbiamo 4 gruppi di esempi che soddisfano A^-:

	Classe C	Altre Classi
Soddisfa X	Y1	E1
Non soddisfa X	Y2	E2

Generalizzazione delle regole

- R copre Y1+E1 esempi di cui E1 erroneamente
- R^- copre Y1+Y2+E1+E2 esempi di cui E1+E2 erronemente
- Data una regola che copre N esempi di cui E erroneamente, e' molto difficile che la sua frazione di errori sia E/N sui casi non visti
- Come nel caso della potatura degli alberi, si usa come stima della frazione di errore il limite superiore dell'intervallo di confidenza per la frazione $U_{CF}(E,N)$ (CF e' dato) che ha una distribuzione di tipo binomiale

Generalizzazione delle regole

- Lo stesso approccio si usa qui:
- La stima dell'errore di R e' $U_{CF}(E1,Y1+E1)$
- La stima dell'errore di R^- e' $U_{CF}(E1+E2,Y1+Y2+E1+E2)$
- Se la stima dell'errore di R^- e' non maggiore di quello di R, allora si puo' cancellare X

Generalizzazione delle regole

- Come cancellare piu' di una condizione?
- In teoria, bisognerebbe considerare ogni possibile sottoinsieme di condizioni
- In pratica si adotta un algoritmo greedy:
  - Finche' ci sono condizioni che si possono cancellare
    - Si cancella la condizione che produce la stima piu' bassa dell'errore
  - Fine ciclo

Generalizzazione delle regole

- Esempio: diagnosi di ipotiroidismo

```
TSH <= 6 : negative (2246.8/1.0)
TSH > 6 :
|   FTI <= 64 :
|   |   TSH measured = f: negative (4.3)
|   |   TSH measured = t:
|   |   |   T4U measured = f:
|   |   |   |   TSH <= 17 : compensated hypothyroid (2.4/0.5)
|   |   |   |   TSH > 17 : primary hypothyroid (2.1/1.1)
|   |   |   T4U measured = t:
|   |   |   |   thyroid surgery = f: primary hypothyroid (59.0/1.0)
|   |   |   |   thyroid surgery = t: negative (3.0/1.0) *
|   FTI > 64 :
|   |   on thyroxine = t: negative (35.2)
|   |   on thyroxine = f:
|   |   |   TSH measured = f: negative (21.2)
|   |   |   TSH measured = t:
|   |   |   |   thyroid surgery = t: negative (3.7)
|   |   |   |   thyroid surgery = f:
|   |   |   |   |   TT4 > 150 : negative (6.1/0.1)
|   |   |   |   |   TT4 <= 150 :
|   |   |   |   |   |   TT4 measured = f: primary hypothyroid (2.8/0.7)
|   |   |   |   |   |   TT4 measured = t: compensated hypothyroid
    (127.4/1.5)
```

Generalizzazione delle regole

- Esempio: regola (indicata con * nell'albero)

If TSH>6 and

FTI<=64 and

TSH measured=t and

T4U measured=t and

tyroid surgery=t

then class negative

- Copre 3 esempi di cui 1 erroneamente Y1=2 E1=1
- $U_{25\%}(1,3)=69\%$

Generalizzazione delle regole

- Stima dell'errore nel caso di rimozione dei vari antecedenti

Condizione cancellata	Y1+Y2	E1+E2	Stima dell'errore
TSH>6	3	1	55%
FTI<=64	6	1	34%
TSH measured = t	2	1	69%
T4U measured = t	2	1	69%
Thyroid surgery = t	3	59	97%

- Si cancella la condizione FTI<=64
- Si ripete il calcolo delle stime degli errori

Generalizzazione delle regole

- Alla fine si ottiene la regola

If thyroid surgey = t then class negative

- Che copre 35 esempi commettendo un solo errore e una stima dell'errore di 7%

Selezione delle regole

- Il processo di generalizzazione e' ripetuto per tutte le regole ottenute da ogni percorso nell'albero di decisione
- Ci possono essere regole duplicate
- Le regole non sono piu'
  - mutuamente esclusive => c'e' bisogno di una tecnica per la risoluzione dei conflitti
  - esaustive => c'e' bisogno di una regola di default che indichi la classificazione dei casi che non sono coperti da nessuna regola

Selezione delle regole

- Risoluzione dei conflitti: se piu' regole coprono un caso, in c4.5 si restiuisce la classificazione della prima regola nell'elenco
- E' necessario ordinare le regole
- c4.5 adotta la seguente strategia: raggruppa insieme le regole per ciascuna classe e poi ordina i gruppi
  - Vantaggio: l'ordine delle regole all'interno del gruppo non e' importante

Selezione delle regole

- All'interno dell'insieme di regole per ciascuna classe, c4.5 cerca di scartare alcune regole in modo da rendere la teoria piu' semplice
- Utilizza il principio della Minimum Description Length (MDL) [Rissanen, 1983]

Principio MDL

- Un Mittente e un Destinatario hanno una copia identica di un insieme di esempi di training ma la copia del Mittente specifica anche la classe di ciascun esempio mentre quella del Destinatario no
- Il Mittente deve comunicare al Destinatario l'informaziona mancante trasmettendo una teoria di classificazione insieme alle eccezioni alla teoria
- Il Principio MDL dice the la miglior teoria derivabile da un insieme di esempi di training e' quella che minimizza il numero di bit necessari a codificare il messaggio totale composto dalla teoria piu' le eccezioni

Selezione di regole

- Sia S un sottoinsieme di tutte le regole per la classe C
- La lunghezza in bit del messaggio e' data da:
  - Per codificare un regola e' necessario specificare ogni condizione nell'antecedente (il conseguente non deve essere codificato perche' e' sempre C)
    - Le condizioni verranno spedite in un certo ordine ma l'ordine non conta
    - Se ci sono x condizioni, ci sono x! ordini
    - Il numero di bit necessari a codificare la regola in un particolare ordine deve essere diminuito di $\log_2$ (x!)

Selezione di regole

– Per codificare l'insieme di regole si sommano i bit per ciascuna regola e si sottraggono i bit relativi all'ordine delle regole (perche' l'ordine non conta)
– Le eccezioni sono codificate indicando quali casi coperti dalle regole di S sono falsi positivi e quali non coperti sono falsi negativi.
  • Falso positivo: un esempio coperto non appartenente alla classe C
  • Falso negativo: un esempio di C non coperto

Selezione di regole

• Se la regola copre r degli n esempi con fp falsi positivi e fn falsi negativi, il numero di bit necessari a codificare le eccezioni e'

$$\log_2\left(\binom{r}{fp}\right)+\log_2\left(\binom{n-r}{fn}\right)$$

• dove

$$\binom{a}{b}=\frac{a!}{b!(a-b)!}$$

• indica il numero di combinazioni di a elementi a b a b

Selezione di regole

- In realta', invece di considerare la somma dei bit per la teoria piu' quelli delle eccezioni si utilizza la formula

  bit per le eccezioni + W * bit per la teoria
- dove W e' un fattore piu' piccolo di 1
- Questo e' fatto perche' solitamente gli schemi di codifica tendono a sovrastimare il numero di bit necessari a codificare la teoria rispetto a quelli per le eccezioni. Questo puo' essere spiegato con il fatto che l'insieme di attributi e' spesso ridondante e teorie differenti sono spesso funzionalmente identiche.
- Di default W vale 0.5 ma il sistema non e' particolarmente sensitivo al valore di W

Selezione di regole

- L'obiettivo e' quindi di trovare un sottoinsieme S delle regole per la classe C che minimizzi il numero di bit per la codifica.
- Problema simile a quello di determinare il sottoinsieme delle condizioni, pero' un algoritmo greedy in questo caso non funziona bene. Si usa invece:
  - Un algoritmo completo se le regole sono poche
  - Il simulated annealing se sono molte

Simulated annealing

- Prendi una regola per la classe C a caso e
  - Includila in S se non c'e' gia' o
  - Rimuovila se c'e' gia'
- Questa azione produce una variazione ΔB del numero totale di bit
  - Se ΔB e' negativo, tieni il cambiamento
  - Se ΔB e' positivo, tieni il cambiamento con probabilita' $e^{-\Delta B/K}$ dove K e' chiamato temperatura
- Riducendo K ad ogni passo, il sistema tende a convergere ad un insieme di regole con una codifica (pressoche') minima

Selezione di regole

- Esempio: si consideri la classe primary hypothyroid
- Per questa classe ci sono 3 regole indicate con i numeri 4, 5 e 7

Teoria		Eccezioni			Costo totale
S	Costo	Falsi positivi	Falsi negativi	Costo	
{}	0,0	0	64	425,8	425,8
{4}	17,1	2	12	116,8	125,3
{5}	19,8	1	6	63,9	73,8
{7}	15,7	1	61	411,8	419,6
{4,5}	35,8	3	5	64,6	82,5
{4,7}	31,7	3	9	97,8	113,6
{5,7}	34,4	2	3	42,1	59,3
{4,5,7}	49,9	4	2	41,0	65,9

Selezione di regole

- L'insieme {5,7} fornisce il costo minimo di codifica

Ordinamento delle regole

- Ogni insieme di regole per una classe commette una serie di errori di tipo falsi positivi
- Conviene mettere per ultime le classi con molti falsi positivi nella speranza che le regole per le classi precedenti coprano (correttamente) gli esempi
- Quindi l'algoritmo e':
  - Teoria:=∅
  - Ripeti
    - Scegli l'insieme con meno falsi posivi e aggiungilo in fondo alla teoria
    - Togli dal training set gli esempi coperti
    - Ricalcola il numero di falsi positivi sugli esempi rimanenti
  - Finche' tutti gli insiemi sono stati selezionati

Ordinamento delle regole

- Esempio:

Classe	Regole generalizzate	Regole selezionate	Esempi coperti	Falsi positivi	Falsi negativi
negative	6	5	2319	2	3
primary	3	2	66	2	3
compensated	2	1	120	0	9

- Le regole per compensated sono messe per prime seguite da quelle per primary e poi per negative

Scelta della regola di default

- c4.5 scegli come classe di default quella piu' frequente nell'insieme degli esempi non coperti da nessuna regola
- Nel caso due o piu' classi abbiamo uguale frequenza, allora sceglie quello con maggiore frequenza nell'insieme totale di esempi
- Nell'esempio ipotiroide, la classe compensated e' quella che ha piu' esempi (8) non coperti da nessuna regola quindi questa diventa la classe di default.

Seconda selezione di regole

- Si esegue una seconda selezione delle regole secondo il seguente algoritmo:
  - Ripeti
    - Considera ciascuna regola in ordine e valuta se, scartandola, si riduce il numero di errori sul training set.
    - Scarta la prima regola che soddisfa la condizione precedente
  - Finche' nessuna regola puo' essere scartata

Esempio: risultato finale

```
Rule 8:
        on thyroxine = f
        thyroid surgery = f
        TSH > 6
        TT4 <= 150
        FTI > 64
        ->  class compensated hypothyroid  [98.9%]

Rule 5:
        thyroid surgery = f
        TSH > 6
        FTI <= 64
        ->  class primary hypothyroid  [95.6%]

Rule 7:
        on thyroxine = f
        TT4 measured = f
        TSH > 6
        ->  class primary hypothyroid  [45.3%]

Rule 1:
        TSH <= 6
        ->  class negative  [99.9%]
```

Esempio: risultato finale

```
Rule 11:
        on thyroxine = t
        FTI > 64
        ->  class negative  [99.5%]

Rule 2:
        TSH measured = f
        ->  class negative  [99.5%]

Rule 9:
        TT4 > 150
        ->  class negative  [99.4%]

Rule 6:
        thyroid surgery = t
        ->  class negative  [92.7%]

Default class: compensated hypothyroid
```


Esempio: risultato finale

- Questo insieme di regole e' piu' semplice dell'albero iniziale ma ha la stessa performance su un insieme di 1258 casi non visti: 8 errori (0,6%).

Uso di c4.5

- Comando per estrarre le regole dai dati:

  c4.5rules –f labor-neg –u

  dove l'opzione –f serve ad indicare il nome dell'insieme di file da esaminare e –u significa che le regole prodotte devono essere testate sui casi del file labor-neg.test

Output: regole di produzione

```
C4.5 [release 8] rule generator Sat May 18 17:05:42 1996
-------------------------------

    Options:
        File stem <labor-neg>
        Rulesets evaluated on unseen cases

Read 40 cases (16 attributes) from labor-neg

------------------
Processing tree 0

Final rules from tree 0:

Rule 6:
        wage increase first year > 2.5
        statutory holidays > 10
        ->  class good  [93.0%]

Rule 5:
        wage increase first year > 4
        ->  class good  [90.6%]
```

Output: regole di produzione

```
Rule 4:
        wage increase first year <= 4
        statutory holidays <= 10
        ->  class bad  [87.1%]

Rule 3:
        wage increase first year <= 2.5
        working hours > 38
        ->  class bad  [79.4%]

Default class: good
```


Valutazione delle regole

```
Evaluation on training data (40 items):

Rule  Size  Error  Used  Wrong            Advantage
----  ----  -----  ----  -----            ---------
   6     2   7.0%    19      0 (0.0%)        0 (0|0)    good
   5     1   9.4%     3      0 (0.0%)        0 (0|0)    good
   4     2  12.9%    10      0 (0.0%)        7 (7|0)    bad
   3     2  20.6%     3      0 (0.0%)        3 (3|0)    bad

Tested 40, errors 1 (2.5%)   <<


          (a)  (b)      <-classified as
         ----  ----
           26           (a): class good
            1    13     (b): class bad
```

Valutazione delle regole (2)

```
Evaluation on test data (17 items):

Rule  Size  Error  Used  Wrong            Advantage
----  ----  -----  ----  -----            ---------
   6     2   7.0%     9      1 (11.1%)       0 (0|0)    good
   5     1   9.4%     3      1 (33.3%)       0 (0|0)    good
   4     2  12.9%     1      0 (0.0%)        1 (1|0)    bad
   3     2  20.6%     2      0 (0.0%)        2 (2|0)    bad

Tested 17, errors 3 (17.6%)   <<


          (a)  (b)      <-classified as
         ----  ----
           11           (a): class good
            3    3      (b): class bad
```


Interpretazione dell'output

- Errror:stima pessimistica dell'errore
- Advantage **A (B|C)**
  - **B**: casi classificati bene dalla regola che verrebbero classificati scorrettamente se la regola non fosse presente
  - **C**: casi classificati male dalla regola che verrebbero classificati correttamente se la regola non fosse presente
  - **A = B-C**: vantaggio fornito dall'includere la regola nell'insieme di regole.

Confronto tra algoritmi per apprendere regole

- Ricerca da specifico a generale o da generale a specifico?
  - Vantaggio di da generale a specifico: c'è un sola ipotesi massimamente generale, molte ipotesi massimamente specifiche
- Ricerca generate and test (CN2, c4.5) oppure example-driven (Find-S, Candidate-Elimination, AQ)?
  - Vantaggio di generate and test: ogni scelta nella ricerca è basata su molti esempi, quindi l'impatto di un esempio rumoroso è basso

Confronto tra algoritmi per apprendere regole

- Quando e come fare post-pruning?
  - Ad esempio: potare ciascuna regola dopo l'apprendimento rimuovendo precondizioni quando questo porti ad un miglioramento della performance su un insieme di esempi distinti da quelli di training
- Quali misure di performance utilizzare?

Software

- CN2 è scaricabile da

http://www.cs.utexas.edu/users/pclark/software.html#cn2

Bibliografia

Clark, P. e Niblett, T. *The CN2 induction algorithm*, Machine Learning 3(4), 1989.

Cervone G., Panait, L.A. and Michalski R.S., *The Development of the AQ20 Learning System and Initial Experiments*, Tenth International Symposium on Intelligent Information Systems, Zakopane, Poland, giugno, 2001. http://www.mli.gmu.edu/papers/IIS-01.ps

Rissanen, J. *A universal prior for integers and estimation by minimum description length*, Annals of Statistics 11(2), 416-431, 1983.