COMPITO DI APPLICAZIONI DI INTELLIGENZA ARTIFICIALE

13 luglio 2004 (Punteggio su 30/30; Tempo 2h)

Esercizio 1 (punti 8)

Dato il seguente training set S:

A1	A2	Classe
1	1	+
1	2	-
2	3	+
2	1	+
2	2	+
2	3	-
1	1	-
?	2	+
1	3	-

a) Si calcoli l'entropia del training set rispetto all'attributo Classe
b) Si calcoli il gain ratio dei due attributi rispetto a questi esempi di training.
c) si costruisca un albero decisionale ad un solo livello per il training set dato, indicando le etichette delle foglie (numero di esempi finiti nella foglia/numero di esempi finiti nella foglia non appartenenti alla classe della foglia).
d) si classifichi l'istanza

1	2

Esercizio 2 (punti 8)

Si consideri il problema di trasportare un carico tramite un carrello ferroviario che può percorrere le tratte tra città connesse. Inizialmente il carico è a milano, già caricato sul carrello ferroviario. Come goal il carico deve arrivare a roma.

Caricamento di un oggetto

```
load(Oggetto,Carrello,Location)
PREC:at(Oggetto,Location), at(Carrello,Location)
ADD LIST: in(Oggetto,Carrello)
DELETE LIST: at(Oggetto,Location)
```

Trasporto

```
drive(Carrello,Location1,Location2)
PREC: at(Carrello,Location1), connected(Location1,Location2)
ADD LIST: at(Carrello,Location2)
DELETE LIST: at(Carrello,Location1)
```

Scaricamento di un oggetto

```
unload(Oggetto,Carrello,Location)
PREC:at(Carrello,Location), in(Oggetto,Carrello)
ADD LIST: at(Oggetto,Location)
DELETE LIST: in(Oggetto,Carrello)
```

Stato iniziale: **`in(carico1,carrello1), at(carrello1,milano)`**
`connected(milano,bologna), connected(bologna,roma)`
Stato goal: **`at(carico1,roma)`**

Si mostrino i passi compiuti dall'algoritmo STRIPS per risolvere il problema. Si mostri UNA SOLA STRADA nello spazio di ricerca che porti a una soluzione.

Esercizio 2 (punti 8)

Ad una festa ci sono sei invitati, identificati con il nome della loro iniziale da A ad F. Ciascuno degli invitati deve essere fatto accomodare ad un tavolo, seguendo le seguenti regole:

- Ci sono 3 tavoli
- Ad ogni tavolo possono essere fatte accomodare al più 2 persone
- Non devono essere messe nello stesso tavolo due persone che si odiano: A odia B, D odia A, E odia F.
- Se possibile, bisogna mettere allo stesso tavolo le persone che si amano (cioè bisogna massimizzare il numero di persone che si amano e che sono sedute allo stesso tavolo). A ama C, E ama D.

Si scriva un programma CLP(FD) che risolve il problema fornito, sapendo che il linguaggio fornito contiene il seguente vincolo:

atmost(+N, ?List, +V): At most N elements of the list List have the value V.
+N An integer
?List A list of domain variables or integers
+V An integer

Esercizio 4 (punti 6)

Si descriva la tecnica di pianificazione come ricerca nello spazio dei piani, evidenziandone i vantaggi rispetto alla pianficazione come ricerca nello spazio degli stati.

SOLUZIONE

Esercizio 1:

a) info(S)=-4/9*$\log_2$ 4/9-5/9*$\log_2$ 5/9=0.991

b)Per caclolare il guadagno dell'attributo A1 non si usa l'entropia calcolata su tutto il training set ma solo sugli esempi che hanno A1 noto (insieme F):

info(F)=-4/8*$\log_2$ 4/8 – 4/8*$\log_2$ 4/8=1

$info_{A1}$(F)=4/8*(-1/4*$\log_2$1/4-3/4*$\log_2$ 3/4)+4/8(-3/4*$\log_2$ 3/4-1/4*$\log_2$ 1/4) =

=0,5*0,811+0,5*0,811=0,811

gain(A1)=8/9*(1-0,811)=0,168

splitinfo(A1)=-4/9*$\log_2$(4/9)-4/9*$\log_2$(4/9)-1/9*$\log_2$(1/9)=1,392

gainratio(A1)=0,168/1,392=0,121

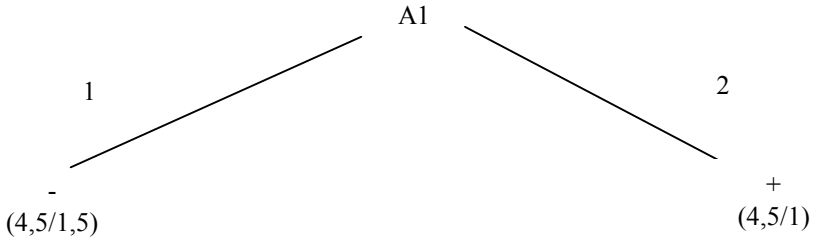
$info_{A2}$(S)=3/9*(-2/3*$\log_2$2/3-1/3*$\log_2$ 1/3)+3/9*(-1/3*$\log_2$ 1/3-2/3*$\log_2$ 2/3)+3/9*(-1/3*$\log_2$ 1/3-2/3*$\log_2$ 2/3)=0,918

gain(A2)=0,991-0,918=0,073

splitinfo(A2)=-3/9*$\log_2$(3/9)-3/9*$\log_2$(3/9)-3/9*$\log_2$(3/9)=1,585

gainratio(A2)=0,073/1,585=0,046

c)



d) l'istanza viene classificata nella foglia di sinistra, quindi appartiene alla classe – con probabilita' 3/4,5=0,667 e alla classe + con probabilità 1,5/4,5=0,333

Esercizio 2

STATO	GOAL
in(carico1,carrello1)	at(carico1,roma)
at(carrello1,milano)	
connected(milano,bologna)	
connected(bologna,roma)	

L'unica azione che ha un effetto che puo' unificare con at(carico1,roma) e' la unload

STATO	GOAL
in(carico1,carrello1)	in(carico1,carrello1)
at(carrello1,milano)	at(carrello1,roma)
connected(milano,bologna)	in(carico1,carrello1) $\wedge$ at(carrello1,roma)
connected(bologna,roma)	**unload(carico1,carrello1,roma)**

in(carico1,carrello1) è già soddisfatto nello stato

STATO	GOAL
in(carico1,carrello1)	
at(carrello1,milano)	at(carrello1,roma)
connected(milano,bologna)	in(carico1,carrello1) $\wedge$ at(carrello1,roma)
connected(bologna,roma)	**unload(carico1,carrello1,roma)**

at(carrello1,roma) può essere soddisfatto tramite l'azione drive

STATO	GOAL
in(carico1,carrello1)	connected(Location1,roma)
at(carrello1,milano)	at(carrello1,Location1)
connected(milano,bologna)	at(carrello1, Location1) $\wedge$ connected(Location1,roma)
connected(bologna,roma)	**drive(carrello1,Location1,roma**)
	in(carico1,carrello1) $\wedge$ at(carrello1,roma)
	unload(carico1,carrello1,roma)

STATO	GOAL
in(carico1,carrello1)	
at(carrello1,milano)	at(carrello1,bologna)
connected(milano,bologna)	at(carrello1, bologna) $\wedge$ connected(bologna,roma)
connected(bologna,roma)	**drive(carrello1,bologna,roma**)
	in(carico1,carrello1) $\wedge$ at(carrello1,roma)
	unload(carico1,carrello1,roma)

at(carrello1,bologna) può essere soddisfatto da drive.

STATO	GOAL
in(carico1,carrello1)	at(carrello1, Location2) ∧ connected(Location2,bologna)
at(carrello1,milano)	**drive(carrello1,Location2,bologna**)
connected(milano,bologna)	at(carrello1, bologna) ∧ connected(bologna,roma)
connected(bologna,roma)	**drive(carrello1,bologna,roma**)
	in(carico1,carrello1) ∧ at(carrello1,roma)
	unload(carico1,carrello1,roma)

la congiunzione di goal è soddisfatta nello stato con Location2 legato a milano

STATO	GOAL
in(carico1,carrello1)	
at(carrello1,milano)	**drive(carrello1,milano,bologna**)
connected(milano,bologna)	at(carrello1, bologna) ∧ connected(bologna,roma)
connected(bologna,roma)	**drive(carrello1,bologna,roma**)
	in(carico1,carrello1) ∧ at(carrello1,roma)
	unload(carico1,carrello1,roma)

Eseguo drive(carrello1,milano,bologna)

STATO	GOAL
in(carico1,carrello1)	at(carrello1, bologna) ∧ connected(bologna,roma)
at(carrello1,bologna)	**drive(carrello1,bologna,roma**)
connected(milano,bologna)	in(carico1,carrello1) ∧ at(carrello1,roma)
connected(bologna,roma)	**unload(carico1,carrello1,roma)**

Ora la congiunzione è soddisfatta

STATO	GOAL
in(carico1,carrello1)	
at(carrello1,bologna)	**drive(carrello1,bologna,roma**)
connected(milano,bologna)	in(carico1,carrello1) ∧ at(carrello1,roma)
connected(bologna,roma)	**unload(carico1,carrello1,roma)**

Eseguo drive(carrello1, bologna, roma)

STATO	GOAL
in(carico1,carrello1)	in(carico1,carrello1) ∧ at(carrello1,roma)
at(carrello1,roma)	**unload(carico1,carrello1,roma)**
connected(milano,bologna)	
connected(bologna,roma)	

Ora la congiunzione è soddisfatta

STATO	GOAL
in(carico1,carrello1)	
at(carrello1,roma)	**unload(carico1,carrello1,roma)**
connected(milano,bologna)	
connected(bologna,roma)	

Eseguo unload (carico1,carrello1,roma)

STATO	GOAL
at(carico1,roma)	
at(carrello1,roma)	
connected(milano,bologna)	
connected(bologna,roma)	

Esercizio 3

Associamo ad ogni ospite una variabile (A,B,C,D,E,F), dobbiamo assegnare un valore che rappresenta il tavolo (da 1 a 3):

```
festa(L):-
    L=[A,B,C,D,E,F],        % Lista degli invitati
    L :: 0..2,              % Numero dei tavoli
    A #\= B,                % Invitati che si odiano
    A #\= D,
    E #\= F,
    atmost(2,L,0),          % Al massimo ci sono 2 invitati per tavolo
    atmost(2,L,1),
    atmost(2,L,2),
                            % Costruisco la funzione obiettivo
```

```
A #= C #<=> Ama1,      % Ama1 vale 1 se A e` nello stesso tavolo di C
E #= D #<=> Ama2,      % Ama2 vale 1 se E e` nello stesso tavolo di D
Fun #= -Ama1 - Ama2,   % Funzione da minimizzare: -Ama1-Ama2
minimize(labeling(L),Fun).
```