Departamento de Ciência de Computadores - FCUP
Segundo Teste de Inteligência Artificial / Sistemas Inteligentes
(Duração: 1 hora)

Data: 24 de Maio de 2017

1) Considere as redes Bayesianas da Figura 1, construídas automaticamente (as tabelas de probabilidade foram omitidas). Qual (ou quais) delas foi (foram) construída(s) de forma a reduzir a complexidade computacional (de tempo e espaço)? (assuma que as variáveis A, B e C são booleanas).

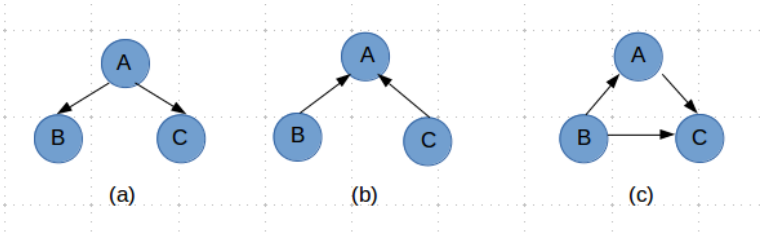

Figura 1: Figura para a pergunta 1

A resposta correta é letra (a). Justificativa: são construídas apenas duas tabelas de probabilidade, com um número mínimo de entradas para cada uma. O facto de economizar memória também ajuda a reduzir o tempo de computação ao fazermos consultas a esta rede.

2) Considere o plano de ordem parcial da Figura 2, onde duas pré-condições já foram resolvidas. O objetivo do plano é levar um agente da sala 1 (room1) para a sala 3 (room3), sabendo que existem portas entre as salas 1 e 2 e entre as salas 2 e 3. Enumere os passos que faria (escolha da próxima pré-condição a ser resolvida e passos seguintes) para levar o algoritmo de geração do plano a encontrar uma solução para este problema.

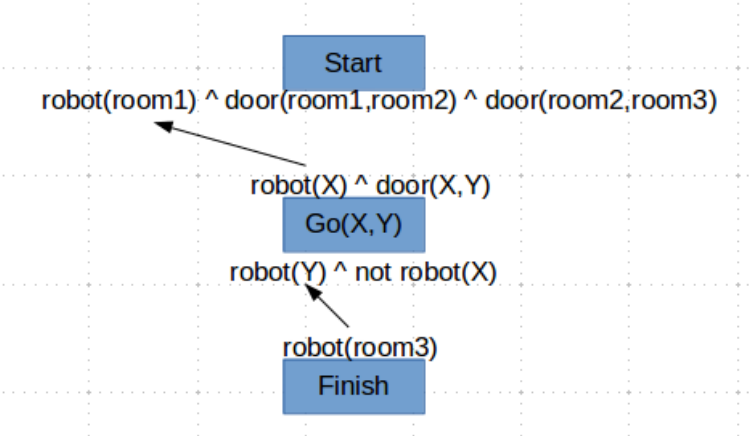

Figura 2: Figura para a pergunta 2

Da forma como o plano está construído, não é possível resolver a única pré-condição que ainda está por resolver: door(room1,room3), porque não há nenhuma ação no plano que satisfaça esta pré-condição (não existe porta entre as salas 1 e 3). Portanto, ocorre um backtracking, que desfaz a unificação feita pela resolução de robot(X) (pré-condição da ação Go/2) com robot(room1) - pós-condiçao do Início. Escolhe-se neste momento resolver door(X,room3) (pré-condição da ação Go/2) com a pós-condição

door(room2,room3) da ação Início. Ao resolver esta pré-condição, a variável X de robot(X) passa a ter valor room2. Falta então resolver esta precondição: robot(room2). Para isto, adiciona-se mais uma ação Go/2 ao plano e continua-se a partir deste ponto, resolvendo as novas pré-condições desta ação Go/2.

3) Num problema de satisfação de restrições com N variáveis e K restrições, onde cada variável pode assumir M valores possíveis, qual seria a complexidade do algoritmo para a resolução deste problema? Justifique.
De acordo com o algoritmo mais usado para resolver problemas de satisfação de restrições, *forward checking*, a complexidade é $O(N \times K \times M)$

4) Utilizando os "perceptrons" que implementam as portas lógicas AND, OR e NOT, desenhe uma rede neuronal que implemente a operação XOR (a saída é 1 quando o número de entradas 1 é ímpar).

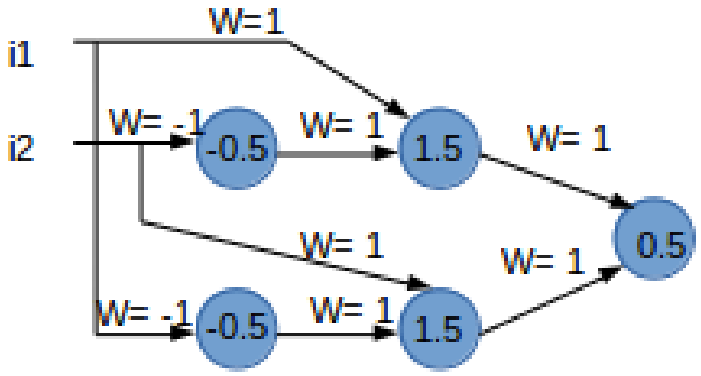

Figura 3: Neural network para implementar um XOR

5) Um algoritmo de indução de árvore de decisão gerou as seguintes tabelas para o cálculo da entropia dos atributos A, B e C, para selecionar o que colocar como raiz da árvore (as linhas correspondem aos valores de cada atributo e as colunas aos valores da variável de classe):

A		B		C	
1	1	1	1	2	2
0	3	2	0	1	2
2	0	1	1		
		0	1		

a) Construa a tabela de dados contendo valores para A, B, C e a variável de classe, que deu origem a estas tabelas.

A tem 3 valores distintos (a1 - quantidade: 2, a2 - quantidade: 3, a3 - quantidade: 2), B tem 4 valores distintos, C tem 2 valores distintos. Há 2 valores para a variável de classe. Os valores da tabela ao

A	B	C	Classe
a1	b1	c1	claro
a1	b1	c1	escuro
a2	b3	c1	escuro
a2	b2	c2	escuro
a2	b2	c2	escuro
a3	b3	c1	claro
a3	b4	c2	claro

lado foram inventados.

b) Qual das variáveis A, B ou C deveria ser escolhida como raiz da árvore?

A variável A porque é a que tem maior poder de discriminação, de acordo com o valor da entropia.

As duas próximas perguntas correspondem à parte prática do teste. Esta parte deve ser respondida apenas por aqueles que não entregaram os dois últimos trabalhos (3 e 4).

(PRÁTICA 3)

P3.1) Para cada uma das duas representações abaixo, que usam a Gramática de Cláusulas Definidas, escreva o código Prolog correspondente:

a) `constante(Dic,I) --> [segunda],{lookup(segunda,Dic,I)}.`

b) `artigo --> [a].`

P3.2) Escreva um código Prolog que, dado um grafo do tipo: `g(a,b). g(a,c). g(a,d). g(b,e). g(b,f). g(c,h). g(h,b). g(d,i). g(d,a).`, uma origem e um destino neste grafo, retorne um caminho ou 'false', sem ciclos.

P3.3) Dada a Gramática de Cláusulas Definidas abaixo e um dicionário de palavras, qual seria o resultado da consulta: `sentenca([a,menina,corre,para,a,floresta],X,Y)`

```
sentenca(sent(FN,FV)) --> fn(FN), fv(FV).
sentenca(sent(FN,FV,C)) --> fn(FN), fv(FV), complemento(C).

fn(fn(A,S)) --> artigo(A), substantivo(S).

fv(verbo(V)) --> [corre].

complemento(compl(prep(para),FN)) --> [para], fn(FN).

artigo(art(a)) --> [a].

substantivo(subs(menina)) --> [menina].

substantivo(subs(floresta)) --> [floresta].
```

(PRÁTICA 4)

P4.1) Seja o algoritmo 1 para indução de árvores de decisão, onde D é um conjunto de instâncias. A linha 9 testa se o melhor atributo é discreto e cria um ramo para cada valor diferente deste atributo.

a) Modifique esta parte do algoritmo para que o nó neste ponto da árvore tenha um número k máximo de ramos menor do que o número de valores diferentes do atributo. Explique em linguagem algorítmica a forma como agrupa os valores discretos e justifique o método escolhido.

b) Modifique esta parte do algoritmo para que este aceite trabalhar com atributos do tipo numérico contínuo. Explique em linguagem algorítmica o tipo de discretização utilizado. Justifique a escolha do método de discretização.

Algorithm 1 Algoritmo para indução de árvores de decisão para a pergunta (P4.1)

```
1: procedure GENERATE_DECISION_TREE(D, attribute_list)
2:     create a node N
3:     if tuples in D are all of the same class C then return N as a leaf node labeled with the class C
4:     end if
5:     if attribute_list is empty then return N as a leaf node labeled with the majority class in D
6:     end if
7:     best_attribute ← attribute_selection(D,attribute_list)
8:     label node N with best_attribute
9:     if best_attribute is discrete-valued then attribute_list ← attribute_list - best_attribute
10:    end if
11:    for each outcome j of best_attribute do
12:        let D_j be the set of data tuples in D satisfying outcome j
13:        if D_j is empty then attach a leaf labeled with the majority class in D to node N
14:        elseattach the node returned by a call to Generate_Decision_Tree(D_j,attribute_list)
15:        end if
16:    end for
17: end procedure
```

P4.2) Suponha que tem a árvore de decisão da Figura 4. Ao testar o exemplo: "Patrons=Full, Hungry=No, Type=Japanese", qual seria a classe esperada: "Yes" ou "No"?

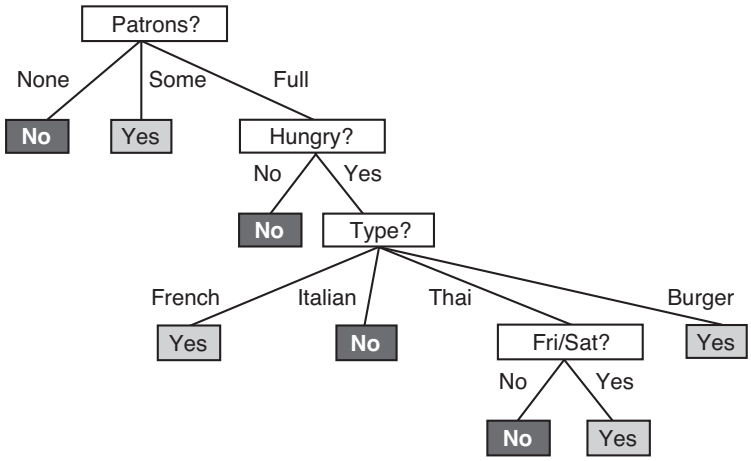

Figura 4: Exemplo de árvore de decisão para a pergunta (P4.2)