

Written exam: Cooperating Intelligent Systems, October 25, 2010

You must achieve at least 50% of the points on this written exam to continue to the oral exam. The total number of points is 53.

No books, mobile phones or calculators are permitted during the exam.

1. Informed search

One of the more important problems in a container terminal is related to the “container stacking problem”. A container stack is a temporary store where containers await further transport by truck, train or vessel. The main efficiency problem for an individual stack is to ensure easy access to containers at the expected time of transfer. Since stacks are “last-in, first-out”, and the cranes (see Figure 1) used to relocate containers within the stack are heavily used, the stacks must be maintained in a state that minimizes on-demand relocations.



Figure 1: A container crane in a harbor.

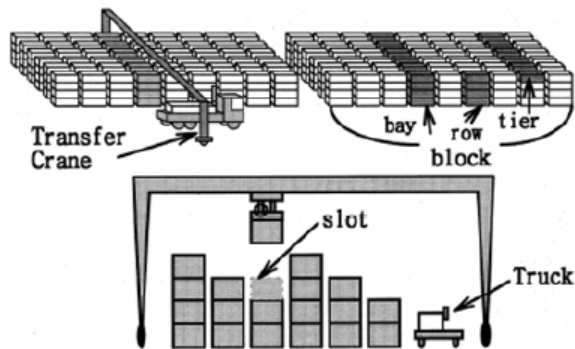


Figure 2: An illustration of the problem domain.

The “container stacking problem” is to restack containers in loading bays (see Figure 2) so that the containers that are marked for the next ship are at the top slots (the top of the row) or do not have any container above them that is not about to be moved out. The problem should be solved quickly (i.e. with the minimum number of moves). An example initial state and an example final state are shown in Figure 3, where the containers that are about to be moved out are marked in red. Note that in the final state can contain two or more containers that are all about to be moved out are stacked on top of each other. It is not allowed to stack more than four containers on top of each other.

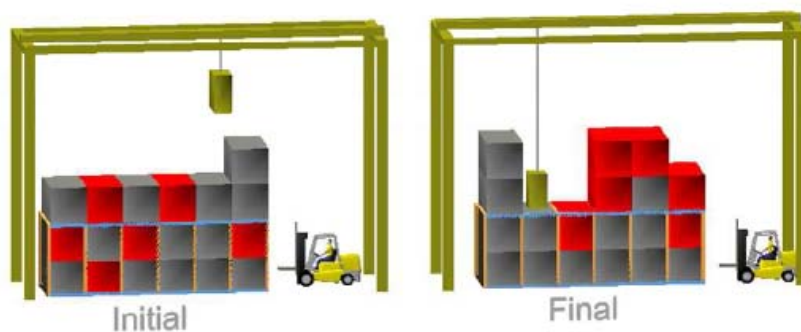


Figure 3: Example initial state (left) and final state after restacking (right). The red containers are the ones that are about to be moved out and should thus be easily accessible.

Solutions

There are six rows of containers in each bay (see Figure 2) and we will restack each bay independently of the other bays.

(a) Formulate a heuristic function for the problem; a heuristic function that can be used with both Greedy-Best-First search and A* search. Explain why it is a suitable heuristic. [2p]

Answer: We need a heuristic that is admissible, i.e. does not overestimate the true number of moves (distance) needed to reach the goal. We separate the containers into “goal containers” and “non-goal containers”; the “goal containers” are those that need to be accessible after the restacking (red containers in Figure 3 and black containers in Figure 4) and the “non-goal containers” are the other containers (non-red containers in Figure 3 and white containers in Figure 4).

Clearly, one admissible heuristic is

$h_1(n)$ = Number of “non-goal” containers on top of a goal container.

$h_1(0) = 5$ for the initial state in Figure 4.

Another admissible heuristic is

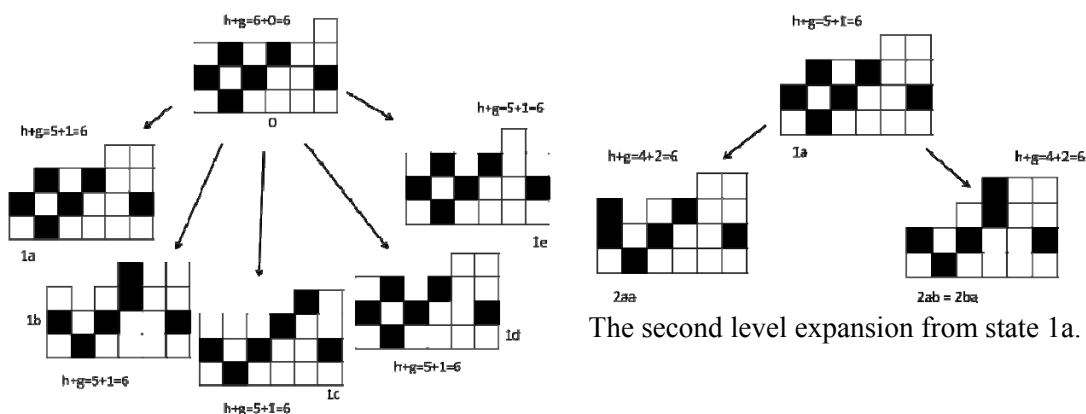
$h_2(n)$ = Number of “non-goal” containers on top of a goal container + number of “goal” containers above a “goal” container if there is at least one “non-goal” container between them.

$h_2(0) = 6$ for the initial state in Figure 4.

Since $h_1(n) \leq h_2(n)$ we use $h_2(n)$. This leads to fewer state expansions.

(b) Show the first levels of expansion when A* searches for a solution starting from the initial state in Figure 4. Explain clearly the expansions. [3p]

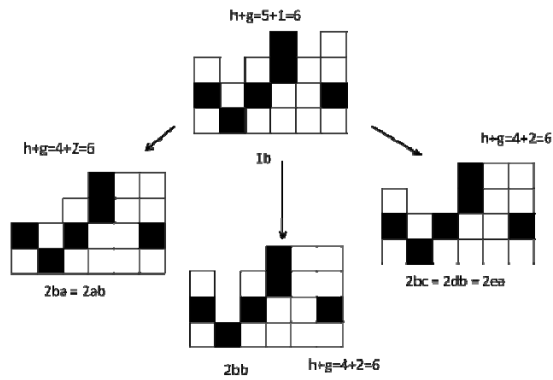
Answer: The figures below show the first two levels of expansion when using A* and the heuristic h_2 . Only states with $h+g = 6$ are shown. All other states have $h+g > 6$.



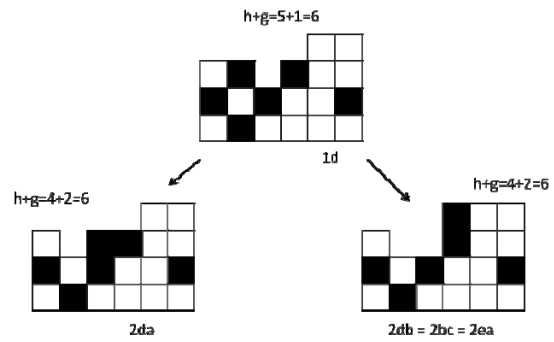
The first level expansion.

The second level expansion from state 1a.

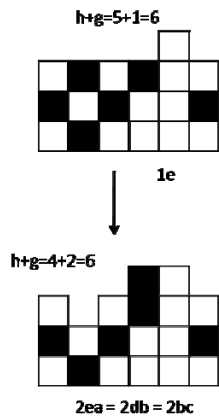
Solutions



The second level expansion from state 1b.



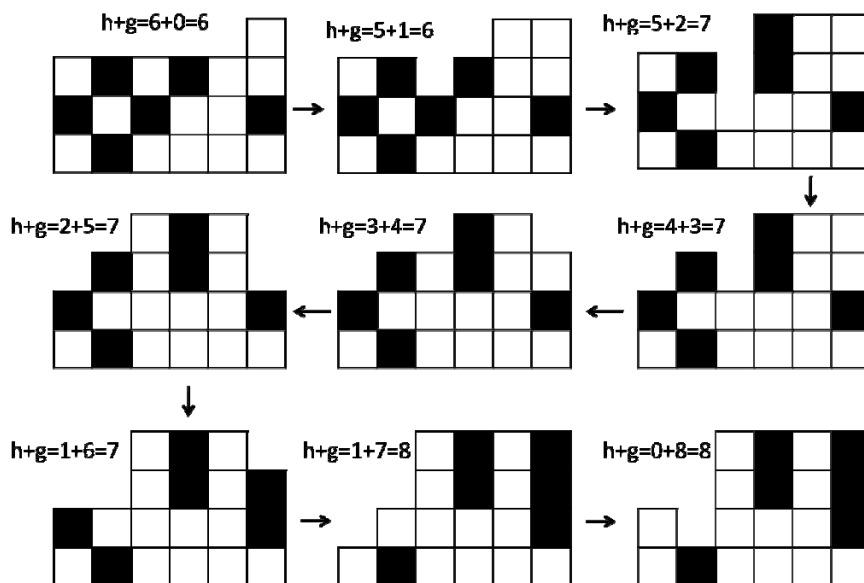
The second level expansion from state 1d (all child states from state 1c have $h+g > 6$).



The second level expansion from state 1e.

(c) Show all the steps from the initial state (Figure 4) to a final state (do not write out all states, just the essential ones). Note: this is a big search list and may take time. It is recommended that you do all the other questions before doing this one. How many moves are needed? [5p]

Answer: This is a bit lengthy answer so we don't show it all here. After expanding all states with $h+g=7$ you see that it is impossible to reach the goal in 7 moves. However, there is at least one solution with 8 moves:



Solutions

Just guessing that the minimum solution is 8 moves will give 1 point.

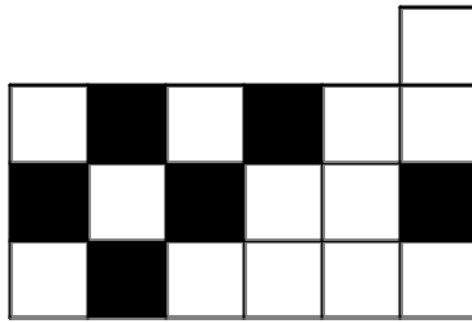


Figure 4: Initial state for the container stacking problem. Black containers are containers that need to be accessible (i.e. not have any white containers above them).

The illustrations in Figure 1, Figure 2 and Figure 3 are taken from “An Artificial Intelligence Planning tool for The Container Stacking Problem” and “A Planning Tool for Minimizing Reshuffles in Container Terminals”, both by Miguel Salido et al. (2009).

2. Game playing

Chinese chess (or Xiangqi 象棋, “Elephant chess”) is a game with 2×16 pieces and 10×10 “squares” (actually points of intersections of lines) where these pieces can be placed.

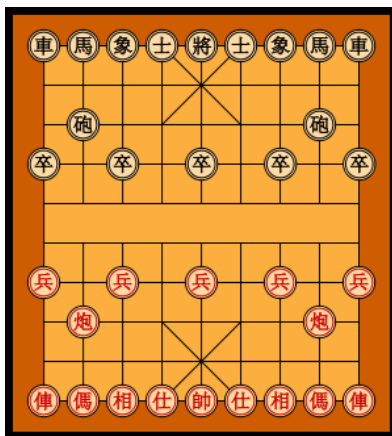


Figure 5: The Chinese chess board and initial positions for the pieces.

Image	Name	No. on each side
	General (King)	1
	Mandarin (Assistant)	2
	Elephant	2
	Horse	2
	Chariot	2
	Cannon	2
	Soldier (Pawn)	5

Figure 6: The pieces in Chinese chess, their names and symbols. The pieces come in two colors and have different signs on them depending on their color.

The Chinese chess board is shown in Figure 5 and the names of the pieces are shown in Figure 6. The pieces are allowed to move in the following ways:

- **General:** One point in any non-diagonal direction. Cannot move outside the castle (the square with diagonal lines on the board, see Figure 5). In addition, the general has the theoretical power of striking along an open file to capture the opposing general. Therefore it is illegal to make any move that leaves your own general on an open file opposite the opposing general, because to do so would be to move into check.
- **Mandarin:** One point in any diagonal direction but is not allowed to move outside the castle.
- **Elephant:** Two points in any diagonal direction. It must move two points, and cannot leap another piece of either color. The elephant is not allowed to cross the river (the “river” is the

Solutions

space created by the two parallel lines that go horizontally across the middle of the board, see Figure 5).

- **Horse:** One point in any non-diagonal direction, followed by one point in a diagonal direction, so that it ends two points away from where it started. This is similar to the knight's move in Western chess, except that the move is blocked by any piece occupying the point at the elbow of the move. Hence it is important to remember that the non-diagonal part of the move comes first.
- **Chariot:** Any number of points in any non-diagonal direction but is not allowed to leap over other pieces (this is just like the rook's move in Western chess).
- **Cannon:** When not capturing, it moves just like the chariot. When capturing, it must leap a single piece of either color before proceeding to the point occupied by the target piece. This intervening piece is called a screen.
- **Soldier:** One point straight forward. After it reaches the opposite river bank (the "other side"), the soldier can move one point forward or directly sideways. The soldier never moves diagonally or backward.

The basic rules are:

- Players take alternate turns. In each turn, a player must make a single move with a single piece. If a piece ends its move on a point occupied by an enemy piece, that piece is captured and permanently removed from play.
- The object of the game is to capture the enemy general. The game is won as soon as one player can make no move that prevents capture of his general. This is checkmate. Stalemate, where one player has no legal move but is not in check, is a win for the last player to move.
- It is illegal to make a move that exposes your general to immediate capture. This is called moving into check.
- It is illegal to avoid defeat or attempt to force a draw by repeating the same series of moves over and over. In particular, perpetual check is not allowed, and the onus is on the attacker to vary his move.

(a) Describe how one would go about to design a program for playing Chinese chess. What are the characteristics of the game, is it deterministic or stochastic, is it fully observable, is it sequential, is it a multi-agent game, is the branching ratio high or low, etc.? [6p]

Answer: This is very much like building a chess-playing program, which we have covered in the lectures. In your answer you are expected to mention the following:

The game is deterministic (no random factor, no dice). The game is fully observable (perfect information). The game is sequential (players take turn making moves). It is a two agent game. The branching ratio is very high (just like in chess). It is a zero-sum game (i.e. one agent wins and the other agent loses the game). There is lots of information available in the literature on old games (just like with chess) so it is good to use book moves, both for the beginning and for the end. There is no chance to search to the end so you need an evaluation function to evaluate non-goal states (describe how such an evaluation function can be designed). You need to implement alpha-beta pruning to reduce the search time. The game does not have much symmetry so you can't utilize this to decrease the search space (there is only a mirror symmetry).

(b) An important technique for faster move evaluation is to use alpha-beta pruning in the search. What is alpha-beta pruning? Which nodes in Figure 7 will not be expanded if we use alpha-beta pruning and expand the nodes from the left (node A is a MIN node). [4p]

Solutions

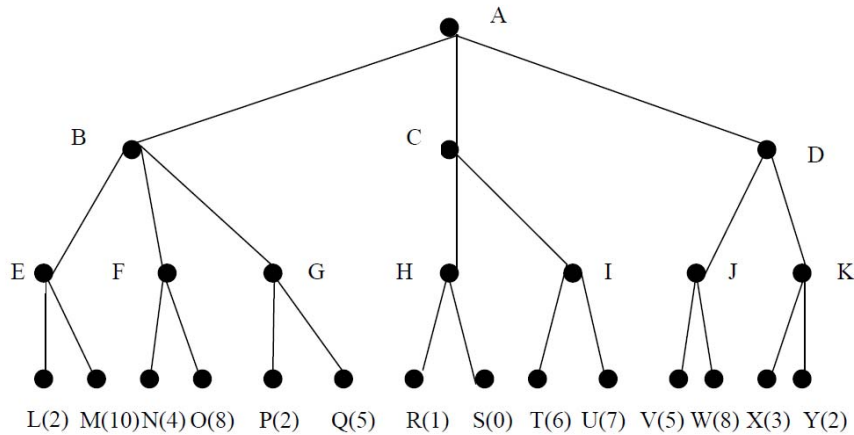
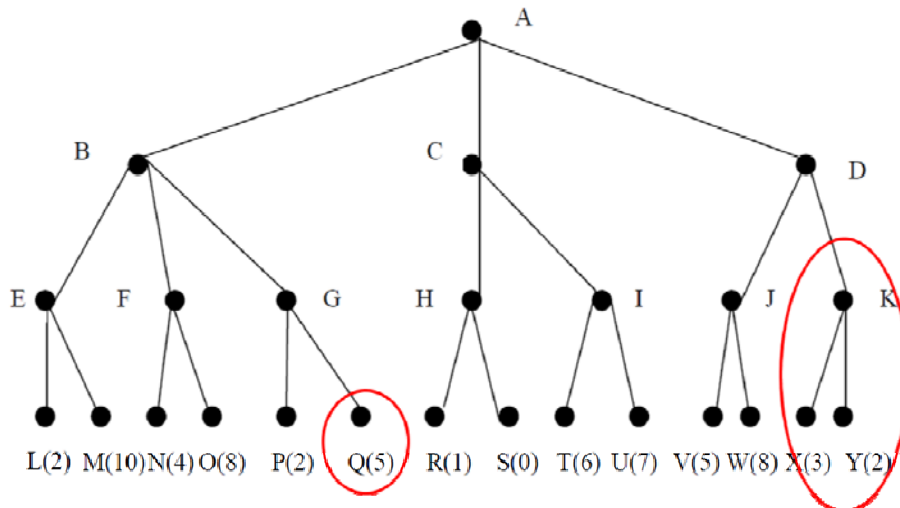


Figure 7: A hypothetical game tree. The initial node is a MIN node (i.e. it is the opponent's move). The lower row shows the utility values for the nodes within parentheses.

Answer:



Note: Images and information about Chinese chess have been collected from "Introduction to Chinese chess" by Donnelly (http://skookumpete.com/chess_intro.htm) and from "Chinese chess" (<https://chessprogramming.wikispaces.com/Chinese+Chess>)

You get 1 point for every correct node and a deduction of 1 point for every extra incorrect node (but never less than 0 points).

3. Logic

(a) Translate these three sentences to propositional logic [3p]

1. Peter will go to the party if and only if Mary does
2. George cannot both study logic and go to the party
3. Either George or Peter must go to the party

Solutions

Answer:

$$1. P \leftrightarrow M$$

$$2. \neg(G \wedge L)$$

$$3. G \vee P$$

(b) Express them in conjunctive normal form (CNF) [3p]

Answer:

$$1a. P \Rightarrow M \equiv \neg P \vee M$$

$$1b. M \Rightarrow P \equiv \neg M \vee P$$

$$2. \neg(G \wedge L) \equiv \neg G \vee \neg L$$

$$3. G \vee P$$

(c) Translate the sentence below to propositional logic, express it in CNF and show, using resolution refutation, if it is entailed from the three sentences above [4p]

George will not study logic unless Mary goes to the party

Answer:

The sentence says that if George studies logic then Mary goes to the party.

$$L \Rightarrow M \equiv \neg L \vee M$$

We combine the negation of this with our knowledge base (statements 1a, 1b, 2 and 3 above).

$$1a. P \Rightarrow M \equiv \neg P \vee M$$

$$1b. M \Rightarrow P \equiv \neg M \vee P$$

$$2. \neg(G \wedge L) \equiv \neg G \vee \neg L$$

$$3. G \vee P$$

$$4. \neg(\neg L \vee M) \equiv L \wedge \neg M$$

Doing “and elimination” for statement 4 we get

$$1a. P \Rightarrow M \equiv \neg P \vee M$$

$$1b. M \Rightarrow P \equiv \neg M \vee P$$

$$2. \neg(G \wedge L) \equiv \neg G \vee \neg L$$

$$3. G \vee P$$

$$4a. L$$

$$4b. \neg M$$

Applying the resolution rule several times gives the following results:

$$1a. P \Rightarrow M \equiv \neg P \vee M$$

$$1b. M \Rightarrow P \equiv \neg M \vee P$$

$$2. \neg(G \wedge L) \equiv \neg G \vee \neg L$$

$$3. G \vee P$$

$$4a. L$$

$$4b. \neg M$$

$$5. \neg P$$

$$6. G$$

$$7. \neg G$$

Here 5 is the resolvent of 4b and 1a; 6 is the resolvent of 5 and 3; 7 is the resolvent of 4a and 2. Finally, 6 and 7 produce the empty set. Hence, the sentence is entailed from the knowledge base.

Use the following propositional symbols:

P = Peter goes to the party;

M = Mary goes to the party;

G = George goes to the party;

L = George studies logic.

4. Bayesian networks

Figure 8 below shows a Bayesian network that describes the relationship between intelligence, hard work and passing an exam. The probabilities in the network are the following:

- $P(\text{Intelligent}) = P(I) = 0.1$
- $P(\text{Study hard}) = P(S) = 0.6$
- $P(\text{Good test taker} \mid \text{Intelligent}) = P(\text{GT} \mid I) = 0.8;$
 $P(\text{GT} \mid \neg I) = 0.5$
- $P(\text{Understands the material} \mid \text{Intelligent and Study hard}) = P(U \mid I, S) = 0.9;$
 $P(U \mid I, \neg S) = 0.3;$
 $P(U \mid \neg I, S) = 0.5;$
 $P(U \mid \neg I, \neg S) = 0.1$
- $P(\text{Pass the exam} \mid \text{Good test taker and Understands the material}) = P(P \mid \text{GT}, U) = 0.9;$
 $P(P \mid \text{GT}, \neg U) = 0.5;$
 $P(P \mid \neg \text{GT}, U) = 0.7;$
 $P(P \mid \neg \text{GT}, \neg U) = 0.3$

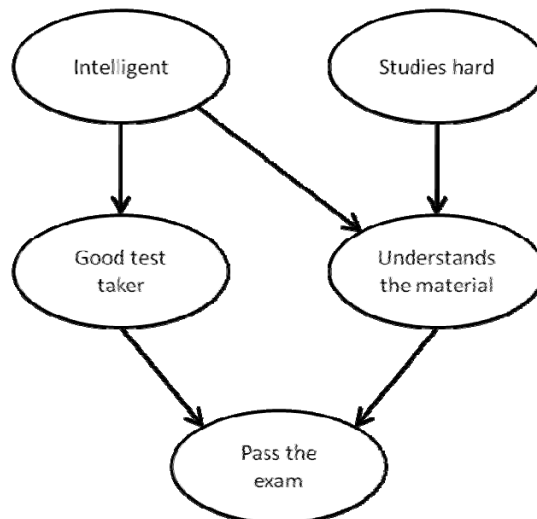


Figure 8: The Bayesian network for passing an exam.

(a) Compute the probability that a student who passes the exam actually understands the material, i.e. compute $P(U | P)$ [6p]

Answer:

$$\begin{aligned}
 P(U | P) &= \frac{P(U, P)}{P(P)} = \\
 &= \frac{\sum_{I, S, GT} P(U, P, I, S, GT)}{\sum_{I, S, GT, U} P(U, P, I, S, GT)} = \\
 &= \frac{\sum_{I, S, GT} P(I)P(S)P(GT | I)P(U | S, I)P(P | GT, U)}{\sum_{I, S, GT, U} P(I)P(S)P(GT | I)P(U | S, I)P(P | GT, U)}
 \end{aligned}$$

Doing the marginalization (the sums) yields, first for the enumerator:

$$\begin{aligned}
 &\sum_{I, S, GT} P(I)P(S)P(GT | I)P(U | S, I)P(P | GT, U) = \\
 &P(I) \sum_{S, GT} P(S)P(GT | I)P(U | S, I)P(P | GT, U) + \\
 &P(-I) \sum_{S, GT} P(S)P(GT | -I)P(U | S, -I)P(P | GT, U) = \\
 &0.1 \cdot \sum_{S, GT} P(S)P(GT | I)P(U | S, I)P(P | GT, U) + \\
 &0.9 \cdot \sum_{S, GT} P(S)P(GT | -I)P(U | S, -I)P(P | GT, U) =
 \end{aligned}$$

Solutions

$$\begin{aligned}
& 0.1 \cdot P(S)P(U | S, I) \sum_{GT} P(GT | I)P(P | GT, U) + \\
& 0.1 \cdot P(\neg S)P(U | \neg S, I) \sum_{GT} P(GT | I)P(P | GT, U) + \\
& 0.9 \cdot P(S)P(U | S, \neg I) \sum_{GT} P(GT | \neg I)P(P | GT, U) + \\
& 0.9 \cdot P(\neg S)P(U | \neg S, \neg I) \sum_{GT} P(GT | \neg I)P(P | GT, U) = \\
& 0.1 \cdot 0.6 \cdot 0.9 \sum_{GT} P(GT | I)P(P | GT, U) + \\
& 0.1 \cdot 0.4 \cdot 0.3 \sum_{GT} P(GT | I)P(P | GT, U) + \\
& 0.9 \cdot 0.6 \cdot 0.5 \sum_{GT} P(GT | \neg I)P(P | GT, U) + \\
& 0.9 \cdot 0.4 \cdot 0.1 \sum_{GT} P(GT | \neg I)P(P | GT, U) = \\
& 0.1 \cdot 0.6 \cdot 0.9 [P(GT | I)P(P | GT, U) + P(\neg GT | I)P(P | \neg GT, U)] + \\
& 0.1 \cdot 0.4 \cdot 0.3 [P(GT | I)P(P | GT, U) + P(\neg GT | I)P(P | \neg GT, U)] + \\
& 0.9 \cdot 0.6 \cdot 0.5 [P(GT | \neg I)P(P | GT, U) + P(\neg GT | \neg I)P(P | \neg GT, U)] + \\
& 0.9 \cdot 0.4 \cdot 0.1 [P(GT | \neg I)P(P | GT, U) + P(\neg GT | \neg I)P(P | \neg GT, U)] = \\
& 0.1 \cdot (0.6 \cdot 0.9 + 0.4 \cdot 0.3) [P(GT | I)P(P | GT, U) + P(\neg GT | I)P(P | \neg GT, U)] + \\
& 0.9 \cdot (0.6 \cdot 0.5 + 0.4 \cdot 0.1) [P(GT | \neg I)P(P | GT, U) + P(\neg GT | \neg I)P(P | \neg GT, U)] = \\
& 0.1 \cdot (0.6 \cdot 0.9 + 0.4 \cdot 0.3) [0.8 \cdot 0.9 + 0.2 \cdot 0.7] + \\
& 0.9 \cdot (0.6 \cdot 0.5 + 0.4 \cdot 0.1) [0.5 \cdot 0.9 + 0.5 \cdot 0.7]
\end{aligned}$$

Which equals $(58 \times 86 + 9 \times 34 \times 80) \times 10^{-5}$ (for the numerator).

The same operations for the denominator yields:

$$\begin{aligned}
& \sum_{I, S, GT, U} P(I)P(S)P(GT | I)P(U | S, I)P(P | GT, U) = \\
& P(I) \sum_{S, GT, U} P(S)P(GT | I)P(U | S, I)P(P | GT, U) + \\
& P(\neg I) \sum_{S, GT, U} P(S)P(GT | \neg I)P(U | S, \neg I)P(P | GT, U) = \\
& 0.1 \cdot \sum_{S, GT, U} P(S)P(GT | I)P(U | S, I)P(P | GT, U) + \\
& 0.9 \cdot \sum_{S, GT, U} P(S)P(GT | \neg I)P(U | S, \neg I)P(P | GT, U) =
\end{aligned}$$

Solutions

$$\begin{aligned}
& 0.1 \cdot P(S) \sum_{GT,U} P(GT | I) P(U | S, I) P(P | GT, U) + \\
& 0.1 \cdot P(\neg S) \sum_{GT,U} P(GT | I) P(U | \neg S, I) P(P | GT, U) + \\
& 0.9 \cdot P(S) \sum_{GT,U} P(GT | \neg I) P(U | S, \neg I) P(P | GT, U) + \\
& 0.9 \cdot P(\neg S) \sum_{GT,U} P(GT | \neg I) P(U | \neg S, \neg I) P(P | GT, U) = \\
& 0.1 \cdot 0.6 \sum_{GT,U} P(GT | I) P(U | S, I) P(P | GT, U) + \\
& 0.1 \cdot 0.4 \sum_{GT,U} P(GT | I) P(U | \neg S, I) P(P | GT, U) + \\
& 0.9 \cdot 0.6 \sum_{GT,U} P(GT | \neg I) P(U | S, \neg I) P(P | GT, U) + \\
& 0.9 \cdot 0.4 \sum_{GT,U} P(GT | \neg I) P(U | \neg S, \neg I) P(P | GT, U) = \\
& 0.1 \cdot 0.6 \left[P(GT | I) \sum_U P(U | S, I) P(P | GT, U) + P(\neg GT | I) \sum_U P(U | S, I) P(P | \neg GT, U) \right] + \\
& 0.1 \cdot 0.4 \left[P(GT | I) \sum_U P(U | \neg S, I) P(P | GT, U) + P(\neg GT | I) \sum_U P(U | \neg S, I) P(P | \neg GT, U) \right] + \\
& 0.9 \cdot 0.6 \left[P(GT | \neg I) \sum_U P(U | S, \neg I) P(P | GT, U) + P(\neg GT | \neg I) \sum_U P(U | S, \neg I) P(P | \neg GT, U) \right] + \\
& 0.9 \cdot 0.4 \left[P(GT | \neg I) \sum_U P(U | \neg S, \neg I) P(P | GT, U) + P(\neg GT | \neg I) \sum_U P(U | \neg S, \neg I) P(P | \neg GT, U) \right] = \\
& 0.1 \cdot 0.6 \left[0.8 \sum_U P(U | S, I) P(P | GT, U) + 0.2 \sum_U P(U | S, I) P(P | \neg GT, U) \right] + \\
& 0.1 \cdot 0.4 \left[0.8 \sum_U P(U | \neg S, I) P(P | GT, U) + 0.2 \sum_U P(U | \neg S, I) P(P | \neg GT, U) \right] + \\
& 0.9 \cdot 0.6 \left[0.5 \sum_U P(U | S, \neg I) P(P | GT, U) + 0.5 \sum_U P(U | S, \neg I) P(P | \neg GT, U) \right] + \\
& 0.9 \cdot 0.4 \left[0.5 \sum_U P(U | \neg S, \neg I) P(P | GT, U) + 0.5 \sum_U P(U | \neg S, \neg I) P(P | \neg GT, U) \right] =
\end{aligned}$$

Solutions

$$\begin{aligned}
 &0.1 \cdot 0.6 [0.8 \cdot (0.9 \cdot 0.9 + 0.1 \cdot 0.5) + 0.2 \cdot (0.9 \cdot 0.7 + 0.1 \cdot 0.3)] + \\
 &0.1 \cdot 0.4 [0.8 \cdot (0.3 \cdot 0.9 + 0.7 \cdot 0.5) + 0.2 \cdot (0.3 \cdot 0.7 + 0.7 \cdot 0.3)] + \\
 &0.9 \cdot 0.6 [0.5 \cdot (0.5 \cdot 0.9 + 0.5 \cdot 0.5) + 0.5 \cdot (0.5 \cdot 0.7 + 0.5 \cdot 0.3)] + \\
 &0.9 \cdot 0.4 [0.5 \cdot (0.1 \cdot 0.9 + 0.9 \cdot 0.5) + 0.5 \cdot (0.1 \cdot 0.7 + 0.9 \cdot 0.3)] = \\
 &6 \cdot (8 \cdot 86 + 2 \cdot 66) \cdot 10^{-5} + 4 \cdot (8 \cdot 62 + 2 \cdot 42) \cdot 10^{-5} + \\
 &54 \cdot (5 \cdot 70 + 5 \cdot 50) \cdot 10^{-5} + 36 \cdot (5 \cdot 54 + 5 \cdot 34) \cdot 10^{-5}
 \end{aligned}$$

The sums within square brackets are computed like this:

$$\begin{aligned}
 &\sum_U P(U | S, I) P(P | GT, U) = \\
 &P(U | S, I) P(P | GT, U) + P(\neg U | S, I) P(P | GT, \neg U) = \\
 &0.9 \cdot 0.9 + 0.1 \cdot 0.5 \\
 &\sum_U P(U | S, I) P(P | \neg GT, U) = \\
 &P(U | S, I) P(P | \neg GT, U) + P(\neg U | S, I) P(P | \neg GT, \neg U) = \\
 &0.9 \cdot 0.7 + 0.1 \cdot 0.3 \\
 &\sum_U P(U | \neg S, I) P(P | GT, U) = \\
 &P(U | \neg S, I) P(P | GT, U) + P(\neg U | \neg S, I) P(P | GT, \neg U) = \\
 &0.3 \cdot 0.9 + 0.7 \cdot 0.5 \\
 &\sum_U P(U | \neg S, I) P(P | \neg GT, U) = \\
 &P(U | \neg S, I) P(P | \neg GT, U) + P(\neg U | \neg S, I) P(P | \neg GT, \neg U) = \\
 &0.3 \cdot 0.7 + 0.7 \cdot 0.3
 \end{aligned}$$

Solutions

$$\begin{aligned}
 & \sum_U P(U | S, \neg I) P(P | GT, U) = \\
 & P(U | S, \neg I) P(P | GT, U) + P(\neg U | S, \neg I) P(P | GT, \neg U) = \\
 & 0.5 \cdot 0.9 + 0.5 \cdot 0.5 \\
 & \sum_U P(U | S, \neg I) P(P | \neg GT, U) = \\
 & P(U | S, \neg I) P(P | \neg GT, U) + P(\neg U | S, \neg I) P(P | \neg GT, \neg U) = \\
 & 0.5 \cdot 0.7 + 0.5 \cdot 0.3 \\
 & \sum_U P(U | \neg S, \neg I) P(P | GT, U) = \\
 & P(U | \neg S, \neg I) P(P | GT, U) + P(\neg U | \neg S, \neg I) P(P | GT, \neg U) = \\
 & 0.1 \cdot 0.9 + 0.9 \cdot 0.5 \\
 & \sum_U P(U | \neg S, \neg I) P(P | \neg GT, U) = \\
 & P(U | \neg S, \neg I) P(P | \neg GT, U) + P(\neg U | \neg S, \neg I) P(P | \neg GT, \neg U) = \\
 & 0.1 \cdot 0.7 + 0.9 \cdot 0.3
 \end{aligned}$$

The denominator is $[6 \times (8 \times 86 + 2 \times 66) + 4 \times (8 \times 62 + 2 \times 42) + 54 \times (5 \times 70 + 5 \times 50) + 36 \times (5 \times 54 + 5 \times 34)] \times 10^{-5}$.

The posterior probability is thus

$$P(U | P) = \frac{29468}{55480} \approx 0.53$$

You get 3 points for setting up the correct expression for $P(U | P)$ and 3 for getting all the numbers right... (after all, it is quite easy to get lost in all the numbers). You get 2 points for the computation if you do reasonable approximations and come to the conclusion that $P(U | P) \approx 50\%$.

(b) Are the following statements true or false in the Bayesian network in Figure 8? [4p]

1. *GT and U are conditionally independent given I.*
2. *P and S are conditionally independent given U.*
3. *P and I are conditionally independent given GT.*
4. *U and I are conditionally independent given S.*

Answer:

1. *GT and U are conditionally independent given I.* TRUE
2. *P and S are conditionally independent given U.* TRUE
3. *P and I are conditionally independent given GT.* FALSE (we must also know U)
4. *U and I are conditionally independent given S.* FALSE

You get 1 point for every correct answer.

5. Machine learning

Master Yoda (Figure 9) is concerned about the number of apprentices who have turned to the Dark Side. He has therefore collected historical data about the apprentices' background and psychological character and decided to build a decision tree to help him identify possible problem apprentices in the future. This historical data is shown in Table 1.

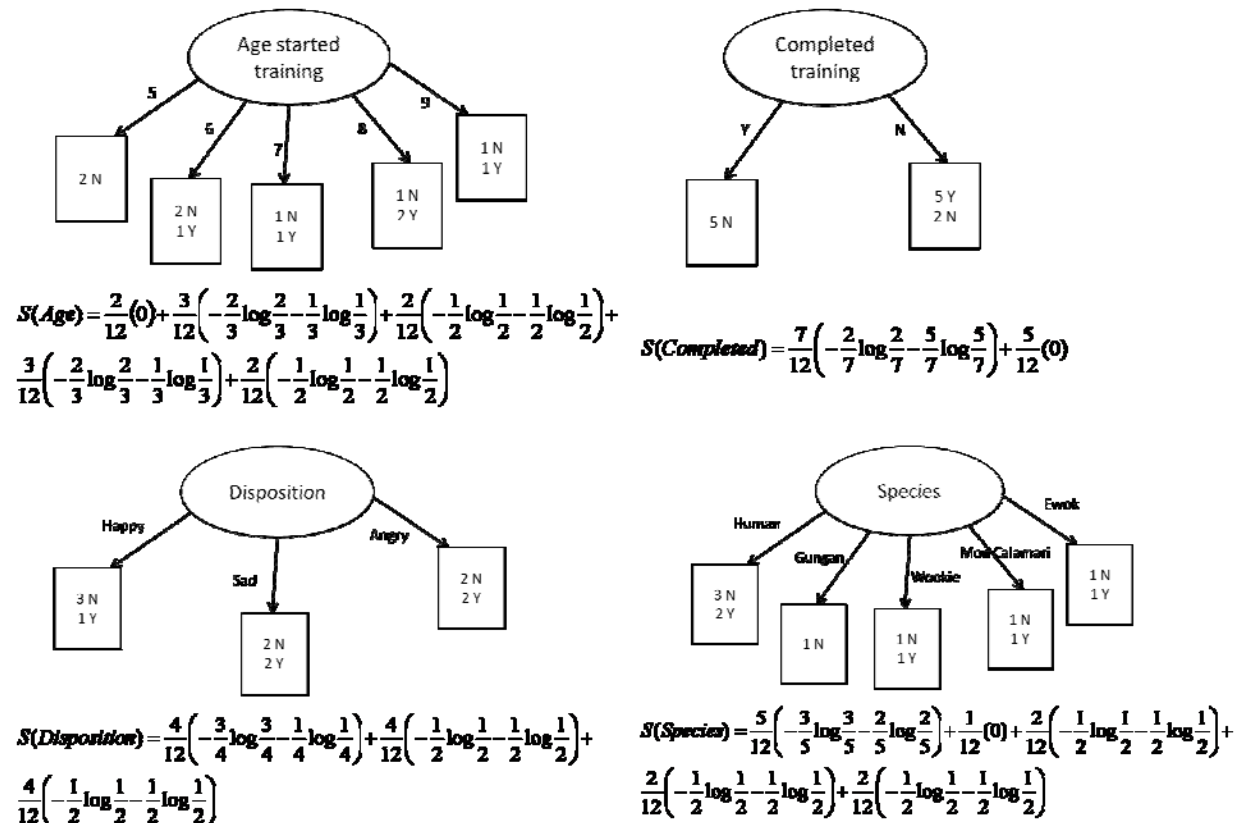


Figure 9: Master Yoda worrying about the high number of Jedi apprentices turning bad...

(a) What is the first feature you should split on if you build a decision tree using the data in Table 1? Explain clearly how you come to this conclusion. [2p]

Answer:

We need to go through each feature and see the resulting entropy. The figures below illustrate them



Then we need to see which of the decisions that leads to the lowest entropy (S). We do this by pairwise comparisons.

$$S(\text{Age}) - S(\text{Completed}) = \dots = \frac{1}{12} \log\left(\frac{2^4 3^4 5^5}{7^7}\right) > 0$$

$$S(\text{Disposition}) - S(\text{Completed}) = \dots = \frac{1}{12} \log\left(\frac{2^{18} 5^5}{3^3 7^7}\right) > 0$$

$$S(\text{Species}) - S(\text{Completed}) = \dots = \frac{1}{12} \log\left(\frac{2^6 5^8}{3 \cdot 7^7}\right) > 0$$

That the differences are positive is determined by checking if the expressions within parentheses are larger or less than one. For example:

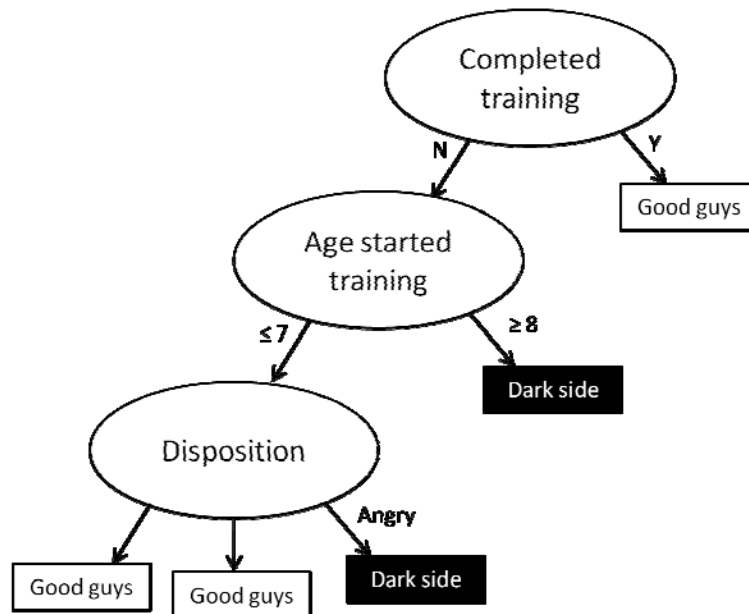
$$\frac{2^4 3^4 5^5}{7^7} = \left(\frac{2 \cdot 3 \cdot 5}{7}\right)^4 \frac{5}{7^3} \approx \frac{4^4}{10 \cdot 7} = \frac{2 \cdot 4^3}{5 \cdot 7} = \frac{2 \cdot 64}{5 \cdot 7} \approx \frac{2 \cdot 9}{5} > 1$$

So, the first decision should be made on *Completed training* since this leads to the lowest entropy.

(b) Show the full decision tree built from the data in Table 1. Explain clearly how you constructed it. [3p]

Answer:

Continuing in the same fashion as for the first decision we get the following final decision tree (in your solution you need to show the computations for the entropies, i.e. determining that the *Age started training* leads to a smaller entropy than any other decision on the second level).



(c) Master Yoda also has three recent Jedi graduates, whose data are shown in Table 2. Who will turn to the Dark Side and who will not? [3p]

Answer: They are all classified as going to the Dark side.

Solutions

(d) Given that Master Yoda does not have a lot of data and that there might be noise in the data (e.g. unmeasured information), how much should you trust the three predictions from the decision tree in (c)? [3p]

Answer: We know that the decision is better the higher in the tree that we are. Both Barbar and Caldar started their training at age 8 so they would have been classified as going to the Dark side higher up in the tree. Therefore, we should trust the predictions about Barbar and Caldar more than the prediction about Ardath.

(e) Suppose you are running a learning experiment on a new algorithm. You have a data set consisting of 25 examples of each of two classes (e.g. Dark Side or not Dark Side). You plan to use leave-one-out cross-validation (i.e. one example is left out for testing each time, which is repeated 50 times). As a baseline, you run your experimental setup on a simple majority classifier. (A majority classifier is given a set of training data and then always outputs the class that is in the majority in the training set, regardless of the input.) You expect the majority classifier to score about 50% on leave-one-out cross-validation, but to your surprise, it scores zero. Explain why. [2p]

Answer: The explanation is quite simple. In the case of equal numbers of two classes will leave-one-out cross-validation produce the worst possible estimate of the generalization performance for a majority classifier. Each time we remove one example, the majority class in the remaining data will be the class that we did not remove the sample from. However, the majority classifier will label the left out example as belonging to the majority class, which will be wrong each time.

Dark Side	Age Started Training	Completed Training	Disposition	Species
0	5	1	Happy	Human
0	9	1	Happy	Gungan
0	6	0	Happy	Wookiee
0	6	1	Sad	Mon Calamari
0	7	0	Sad	Human
0	8	1	Angry	Human
0	5	1	Angry	Ewok
1	9	0	Happy	Ewok
1	8	0	Sad	Human
1	8	0	Sad	Human
1	6	0	Angry	Wookiee
1	7	0	Angry	Mon Calamari

Table 1: Master Yoda's historical data on twelve Jedi apprentices.

Name	Age Started Training	Completed Training	Disposition	Species
Ardath	5	0	Angry	Human
Barbar	8	0	Angry	Gungan
Caldar	8	0	Happy	Mon Calamari

Table 2: Master Yoda's three recent Jedi apprentices that he wants to label with the decision tree.