CS 188 Fall 1999 — Introduction to AI — Stuart Russell

Final solutions

1. (10 pts.) True/False

(a) (2) False; a feedforward network has no internal state and hence no memory.

(b) (2) True; both return the “leftmost” among the shallowest solutions.

(c) (2) True; although the solution to an MDP is a policy rather than just the solution path returned by A*, the rest of the policy besides the solution path is irrelevant because those states are never reached.

(d) (2) True; a neural net with enough hidden nodes can represent any Boolean function.

(e) (2) False; the entailment is the other way.

2. (18 pts.) Logic *True/false*:

(a) (3) True; otherwise we can assign each distinct literal to be false and falsify the clause.

(b) (3) False; $Q(w, A)$ could only resolve against the negative literal $\neg Q(x, F(x))$, leaving a positive literal.

(c) (4) True; $C_1 \models C_1\sigma$ for any σ; if $C_1\sigma \subset C_2$ then $C_1\sigma \models C_2$, by the semantics of disjunction.

(d) (4) False; the dreaded $\exists \ldots \Rightarrow \ldots$.

(e) (4) True; the search tree is linear and finite, and resolution is complete.

3. (14 pts.) Planning and MDPs

(a) (2) $Op(\text{Action}{:}TurnOn(b), \text{Precond}{:}Off(b), \text{Effect}{:}On(b)$ $\quad Op(\text{Action}{:}TurnOff(b), \text{Precond}{:}On(b), \text{Effect}{:}$

(b) (2) Start with postconditions $Off(1)$, $Off(2)$, $Off(3)$ and End with preconditions $On(2)$ and $On(3)$.

(c) (5) The open conditions are $On(2)$ and $On(3)$. These are not achieved by Start so a new step must be added. $TurnOn(2)$ and $TurnOn(3)$ are added with preconditions $Off(2)$ and $Off(3)$. These are achieved by causal links from Start.

(d) (4) An MDP requires the following: **states** are all 8 settings of the three bits; **actions** are all applicable $TurnOn$ and $TurnOff$ actions in each state (3 actions per state but the 2 goal states are absorbing); **rewards** are +ve (say +1) for goal states, -ve for others to ensure shortest solution; **transition model** is deterministic: $Turnon(b)$ turns the bit b on with probability 1 where applicable.

(e) (1) Just need to remember a policy for *all* states: if bit 2 is off, turn it on; if bit 3 is off, turn it on.

4. (16 pts.) Probabilistic inference

(a) (3) (ii) is asserted, by the local semantics of BNs: a node is conditionally independent of its nondescendants given its parents. (i) is not asserted since H and S are linked by an arc. (iii) is not asserted by the structure alone, because arcs do not *deny* independence. (The CPTs can deny it, however.)

(b) (3) $P(h, s, \neg p, \neg e) = P(h)P(s|h)P(\neg p|h, s)P(\neg e|\neg p) = 0.1 \times 0.3 \times 0.1 \times 0.9 = 0.00027$

(c) (4) Probably the simplest way to do this is to construct the part of the full joint for H true (8 rows) and then add up. The following is the enumeration algorithm:

$$\begin{aligned} P(E|h) &= \alpha P(h) \sum sP(s|h) \sum pP(p|h, s)P(E|p) \\ &= \alpha\ 0.1[0.3 \times (0.9\langle 0.6, 0.4\rangle + 0.1\langle 0.1, 0.9\rangle) + 0.7 \times (0.5\langle 0.6, 0.4\rangle + 0.5\langle 0.1, 0.9\rangle)] = \langle 0.41, 0.59\rangle \end{aligned}$$

(d) (6) Let’s assume honesty doesn’t influence fundraising ability, but slickness does. Funds support advertising which increases popularity, but do not directly affect electability otherwise. So L should be a child of S and parent of P. We would need a CPT for $P(L|S)$ and an augmented CPT $P(P|H, S, L)$. Any CPTs refelcting the abovementioned influences will do.

5. (10 pts.) Vision

(a) (4) (i) A appears bigger and cars are usually roughly similar in size; (ii) since A and B are both on the same horizontal plane and B appears *above* A, it *must* be further away.

(b) (6) A, B, C can be viewed in stereo and hence their depths can be measured, allowing the viewer to determine that B is nearest, A and C are equidistant and slightly further away. Neither D nor E can be seen by both cameras, so stereo cannot be used. However, because the bottle *occludes* D from Y and E from X, D and E must be further away than A, B, C, but their relative depths cannot be determined.

6. (12+7 pts.) Robotics

(a) (7) See Fig. 1(a).

(b) (7 *extra credit*) See Fig. 1(b). Boundaries are loci of arm–obstacle contact, e.g.:
– End of arm against left wall: $x = \cos\theta \;\Rightarrow\;$ boundary 1.
– End of arm against top barrier: $2 - x = \cos\theta$ for $\theta \in [\pi/6, \pi/2] \;\Rightarrow\;$ boundary 2.
– Side of arm against top doorpost: $2 - x = 0.5 \cot\theta$ for $\theta \in [\pi/6, \pi/2] \;\Rightarrow\;$ boundary 3.

(c) (5) An ideal robot could move to $x = 1$, rotate to $\theta = 0$, move to $x = 3$, rotate to $\theta = \pi/2$. Any real robot would either err on the $x = 1$, so that the rotation sticks against the left wall or the barrier, or it err on $\theta = 0$, so that the slide to $x = 3$ would jam the arm against the barrier. The solution is to use force feedback: move to $1 < x < 2$; rotate until contact with the barrier; move in $-x$ direction while rotating to maintain sliding contact with barrier until contact is lost (or contact on the opposite side of the arm). Now the arm is in the doorway: slide to $x > 3$ maintaining contact with lower doorpost; as soon as this is lost, rotate to $\theta = \pi/2$, slide until motion stopped by wall.

7. (20 pts.) Learning A *1–decision list* or 1-DL is a decision tree with Boolean inputs in which at least one branch from every attribute test leads immediately to a leaf (obviously the final test leads to two leaves).

(a) (3) Linear tree with attributes a_1, a_2, a_3; leaves are T, T, T, F.

(b) (3) Need to specify 4 weights. By symmetry, w_1, w_2, w_3 are the same; say 2. The "lowest" input requiring +1 output is (say) +1, –1, –1, giving a weighted sum of –2. The input requiring output of –1 is –1, –1, –1, giving a weighted sum of –6. Hence a weight $w_0 = --4$ nicely separates the +ve and -ve cases.

(c) (6) Intuitively, the root of the decision list is the most important, so has the highest weight. If a true attribute requires a false output, then its weight must be negative. Hence A=3, B=2, C=1.

(d) (5) (i) Perceptron learning converges when the data can be represented by the perceptron (see book). (ii) For any DL, there is an equivalent perceptron? Generalizing from the examples in part (c): Essentially, the kth attribute (out of n) along the DL has a weight of $\pm 2^{n-k+1}$, and its sign is determined by the parity of the associated leaf. Then the bias weight w_0 is set so as to give the right answer for the final leaf; this can always be done.

(e) (3) A decision tree can represent any Boolean function including XOR. Perceptrons cannot represent XOR.

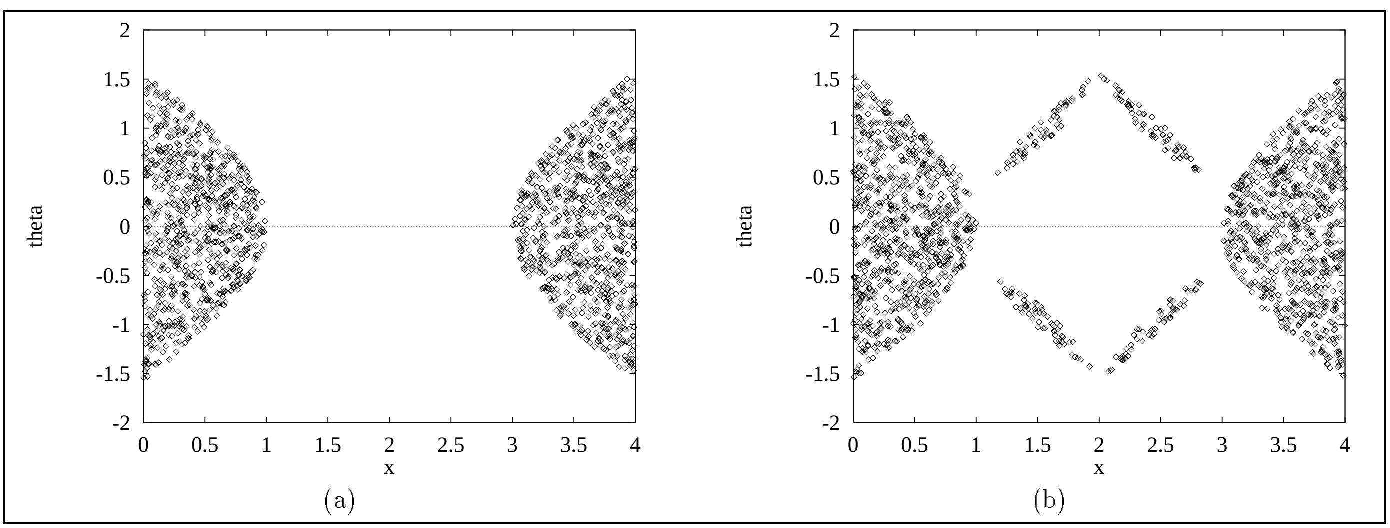

Fig. 1: C-spaces for 6(a) and (b).