CS 188 Spring 2002 | Introduction to AI Stuart Russell

Final Solutions

1. (12 pts.) True/False

(a) (2) False—meaning can depend on context, and some sentences are ambiguous.

(b) (2) True—HMMs are a special case of DBNs.

(c) (2) True—images are scale-invariant

(d) (2) True—a self-resolving clause contains complementary literals, each of which remains in the resolvent.

(e) (2) True—a corollary of the theorem stating that POP is sound and complete.

(f) (2) True—a positive literal is a Horn clause but is falsified by such a model.

2. (15 pts.) Logic

(a) (2) A reasonable translation is, "If two people speak the same language then they understand each other." One could quibble about whether this covers the case where $x = y$.

(b) (3) $Understands(x, y)$ means that x understands y; $Friend(x, y)$ means x is a friend of y.

i. $\forall x, y\ Understands(x, y) \Rightarrow Friend(x, y)$.

ii. $\forall x, y, z\ Friend(x, y) \land Friend(y, z) \Rightarrow Friend(x, z)$

(c) (5) Let the KB contain the following sentences, all Horn clauses:
A1: $\forall x, y, l\ Speaks(x, l) \land Speaks(y, l) \Rightarrow Understands(x, y)$
A2: $\forall x, y, l\ Speaks(x, l) \land Speaks(y, l) \Rightarrow Understands(y, x)$
B1: $\forall x, y\ Understands(x, y) \Rightarrow Friend(x, y)$
B2: $\forall x, y, z\ Friend(x, y) \land Friend(y, z) \Rightarrow Friend(x, z)$
C1: $Speaks(Ann, French)$
C2: $Speaks(Bob, French)$
C3: $Speaks(Bob, German)$
C4: $Speaks(Cal, German)$
We will prove $Friend(Ann, Cal)$ using forward chaining.

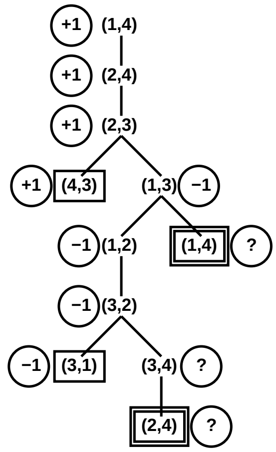
- FC on A1, premises C1, C2, $\{x/Ann, y/Bob, l/French\}$, P1: $Understands(Ann, Bob)$.
- FC on A1, premises C3, C4, $\{x/Bob, y/Cal, l/German\}$, P2: $Understands(Bob, Cal)$.
- FC on B1, premise P1, $\{x/Ann, y/Bob\}$, P3: $Friend(Ann, Bob)$.
- FC on B1, premise P2, $\{x/Bob, y/Cal\}$, P4: $Friend(Bob, Cal)$.
- FC on B2, premises P3, P4, $\{x/Ann, y/Bob, z/Cal\}$, P5: $Friend(Ann, Cal)$.

(d) (5) We need to show that $D \models A$, i.e., $D \land \neg A$ yields a contradiction. The CNF KB is
Q1: $\neg S(x, l) \lor \neg S(y, l) \lor U(x, y)$
Q2: $S(A, F)$
Q3: $S(B, F)$
Q4: $(\neg U(A, B) \lor \neg U(B, A))$
We will prove a contradiction using resolution:

- Resolving Q1, Q2, $\{x/A, l/F\}$, gives Q5: $\neg S(y, F) \lor U(A, y)$
- Resolving Q5, Q3, $\{y/B\}$, gives Q6: $U(A, B)$
- Resolving Q1, Q2, $\{y/A, l/F\}$, gives Q7: $\neg S(x, F) \lor U(x, A)$
- Resolving Q7, Q3, $\{x/B\}$, gives Q8: $U(B, A)$
- Resolving Q6, Q4, $\{\}$, gives Q9: $\neg U(B, A)$
- Resolving Q8, Q9, $\{\}$, gives the empty clause.

3. (14 pts.) Games

(a) (5) Here is the game tree, complete with annotations of all backed-up values.



(b) (5) The "?" values are handled by assuming that an agent with a choice between winning the game and entering a "?" state will always choose the win. That is, min(–1,?) is –1 and max(+1,?) is +1. If all successors are "?", the backed-up value is "?".

(c) (5) Standard minimax is depth-first and would go into an infinite loop. It can be fixed by comparing the current state against the stack, and if it occurs on the stack then returning a "?" value. Propagation of "?" values is handled as above. It does not always work perfectly, because some games have terminal states with differing values (e.g., draws or wins of different degrees). It is not clear how to handle these in the propagation process. There are more complex methods that determine whether the node has a well-determined value (as in this example) or is really an infinite loop in that both players prefer to stay in the loop (or have no choice). In such a case, the loop state can be labelled as a draw.

4. (12 pts.) MDPs and Games

(a) (5) $U_A(s) = R(s) + \max_a \sum_a P(s'|a,s)U_B(s')$ and $U_B(s) = R(s) + \min_a \sum_a P(s'|a,s)U_A(s')$

(b) (5) We simply turn each of them into a Bellman update and apply them in alternation, applying each to all states simultaneously.

	(1,4)	(2,4)	(3,4)	(1,3)	(2,3)	(4,3)	(1,2)	(3,2)	(4,2)	(2,1)	(3,1)
U_A	0	0	0	0	0	**+1**	0	0	**+1**	**–1**	**–1**
U_B	0	0	0	0	–1	**+1**	0	–1	**+1**	**–1**	**–1**
U_A	0	0	0	–1	+1	**+1**	–1	+1	**+1**	**–1**	**–1**

(c) (2) The utility vector for one player is the same as the previous utility vector *for the same player* (i.e., two steps earlier). (Note that typically U_A and U_B are not the same in equilibrium.)

5. (22 pts.) Statistical learning, Bayes nets

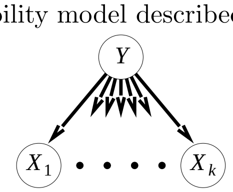
(a) (2) The probability of a positive example is π and of a negative example is $(1-\pi)$, and the data are independent, so the probability of the data is $\pi^p(1-\pi)^n$

(b) (3) We have $L = p \log \pi + n \log(1-\pi)$; if the derivative is zero, we have

$$\frac{\partial L}{\partial \pi} = \frac{p}{\pi} - \frac{n}{1-\pi}$$

so the ML value is $\pi = p/(p+n)$, i.e., the proportion of positive eamples in the data.

(c) (2) This is the "naive Bayes" probability model described in AIMA2e Ch. 13.



(d) (4) The likelihood of a single instance is a product of terms. For a positive example, π times α_i for each true attribute and $(1-\alpha_i)$ for each negative attribute; for a negative example, $(1-\pi)$ times β_i for each true attribute and $(1-\beta_i)$ for each negative attribute. Over the whole data set, the likelihood is $\pi^p(1-\pi)^n \prod_i \alpha_i^{p_i^+}(1-\alpha_i)^{n_i^+}\beta_i^{p_i^-}(1-\beta_i)^{n_i^-}$.

(e) (5) The log likelihood is

$$L = p\log\pi + n\log(1-\pi) + \sum_i p_i^+ \log\alpha_i + n_i^+ \log(1-\alpha_i) + p_i^- \log\beta_i + n_i^- \log(1-\beta_i)$$

Setting the derivatives w.r.t. α_i and β_i to zero, we have

$$\frac{\partial L}{\partial \alpha_i} = \frac{p_i^+}{\alpha_i} - \frac{n_i^+}{1-\alpha_i} = 0 \quad \text{and} \quad \frac{\partial L}{\partial \beta_i} = \frac{p_i^-}{\beta_i} - \frac{n_i^-}{1-\beta_i} = 0$$

giving $\alpha_i = p_i^+/(p_i^+ + n_i^+)$, i.e., the fraction of cases where X_i is true given Y is true, and $\beta_i = p_i^-/(p_i^- + n_i^-)$, i.e., the fraction of cases where X_i is true given Y is false.

(f) (3) In the data set we have $p=2$, $n=2$, $p_i^+=1$, $n_i^+=1$, $p_i^-=1$, $n_i^-=1$. From our formulæ, we obtain $\pi=\alpha_1=\alpha_2=\beta_1=\beta_2=0.5$.

(g) (2) Each example is predicted to be positive with probability 0.5.

(h) (2) The data describe an XOR function. A naive Bayes model is unable to represent the function, just as a one-layer perceptron cannot represent it. The naive Bayes model is discrete rather than continuous, so we cannot map one directly onto the other, but clearly the naive Bayes model with only a linear number of parameters cannot represent the doubly exponential number of Boolean functions.

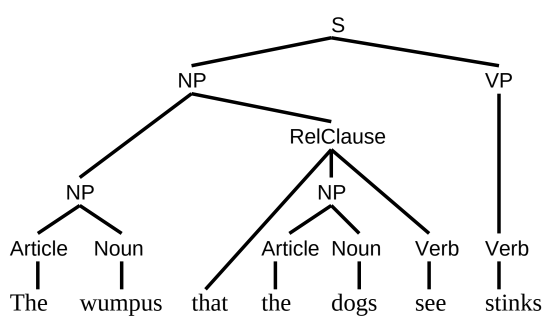
6. (15 pts.) Natural language

(a) (3) (i) is not generated (adjectives may not modify nouns).
(ii) and (iii) are generated (although the parse tree does not reflect any semblance of English grammar).

(b) (4) In "that the dogs see", the object of the sentence is missing, so we need a rule describing transitive sentences with a missing object. The following will do, although it is far from perfect:

RelClause → ***that*** *NP Verb*

(c) (4)



(d) (4) To avoid the recursion that produces the illegal sentences, we have to allow "thatless" relative clauses only with a special kind of noun phrase that does itself not allow "thatless" relative clauses:

NP → *NP2 ThatlessRelClause*

ThatlessRelClause → *NP2 Verb*

where *NP2* is defined as a restricted *NP* that does not allow relative clauses.

7. (10 pts.) Robotics

(a) (5) The configuration space must have a θ_1-obstacle at π, and must disallow small values of θ_1 and θ_2. Space (a) is the only one that does this.

(b) (5) The initial configuration is $(\pi/4, \pi/2)$. The goal configuration is $(7\pi/4, 0)$. It can be reached by, e.g., increasing θ_2 to π, decreasing θ_1 to $6\pi/4$, decreasing θ_2 to 0, then increasing θ_1 to $7\pi/4$.