Question 4. Bayes Nets

A professor is heading back into the lab late at night and is trying to guess whether his student is working on their paper. He has noticed that whenever this student is working, their car is in the garage and there is music playing in the lab. Let W, C, M be three binary random variables corresponding to the events "student Working on paper", "Car in parking lot", and "Music playing". Through careful observation, the professor has determined the following conditional probabilities:

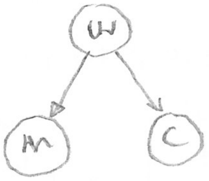
$$(1) \quad P(C|W,M) = P(C|W) = \begin{array}{c|cc} & W=1 & W=0 \\ \hline C=1 & 0.8 & 0.1 \\ C=0 & 0.2 & 0.9 \end{array}$$

$$(2) \quad P(M|W) = P(C|W)$$

$$(3) \quad P(W=1) = 0.8$$

Note that equation (2) says that C and M have the same conditional distribution (e.g. $P(C=1|W=1) = P(M=1|W=1) = 0.8$.)

(a) Draw a Bayesian network model (i.e. a directed graphical model) for this problem.



(b) Compute the marginal distributions $P(C)$ and $P(M)$, before any evidence is availble

$P(C,W) = P(C|W)P(W) = P(M|W)P(W) = P(M,W)$ so joint pdf's also same

$$\overset{c\backslash w}{=} \begin{bmatrix} 0.64 & 0.02 \\ 0.16 & 0.18 \end{bmatrix} \begin{matrix} \rightarrow \\ \rightarrow \end{matrix} \begin{bmatrix} 0.66 \\ 0.34 \end{bmatrix} \begin{matrix} C=1 \\ C=0 \end{matrix}$$

↑ P(C) and P(M)

(c) Upon entering the garage, the professor notices that the student's car is parked there. Calculate $P(W|C=1)$.

Bayes Rule:

$$P(W|C=1) = \frac{P(C=1|W)\,P(W)}{P(C=1)} = \underset{\substack{\uparrow \\ P(C=1) \\ \text{(from} \\ \text{marginal)}}}{\frac{1}{0.66}} \times \underset{\substack{\uparrow \qquad\quad \uparrow \\ P(W) \quad P(C=1|W)}}{\begin{bmatrix} 0.8 \times 0.8 \\ 0.2 \times 0.1 \end{bmatrix}} = \begin{bmatrix} 0.97 \\ 0.03 \end{bmatrix} \begin{matrix} W=1 \\ W=0 \end{matrix}$$

(d) Calculate the updated probability that music is playing in the lab

We need to take C=1 into account

$$P(M,W|C=1) = P(M|W,C=1)\,P(W|C=1) = P(M|W)\,P(W|C=1)$$

$$= \boxed{\begin{array}{c|cc} C \backslash W & 1 & 0 \\ \hline 1 & 0.8 & 0.1 \\ 0 & 0.2 & 0.9 \end{array}} * \overset{W|C=1}{\begin{bmatrix} 0.97 \\ 0.03 \end{bmatrix}} \begin{matrix} 1 \\ 0 \end{matrix}$$

$$= \begin{bmatrix} 0.8 \times 0.97 & 0.1 \times 0.03 \\ 0.2 \times 0.97 & 0.9 \times 0.03 \end{bmatrix} = \begin{array}{c|cc} M \backslash W & 1 & 0 \\ \hline 1 & 0.78 & 0.003 \\ 0 & 0.19 & 0.027 \end{array} \begin{matrix} \rightarrow \\ \rightarrow \end{matrix} \overset{P(M|C=1)}{\boxed{\begin{matrix} 0.783 \\ 0.217 \end{matrix}}} \begin{matrix} 1 \\ 0 \end{matrix}$$

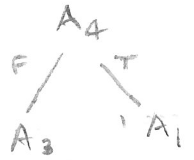
Question 5. Decision Tree Learning

Consider the following dataset for training a decision tree:

	A_1	A_2	A_3	A_4	Output
1	1	1	1	0	1
2	1	0	1	0	1
3	0	0	1	1	1
4	0	0	$\boxed{0}$	0	1
5	1	0	0	1	0
6	1	1	1	1	0
7	1	0	0	0	0
8	0	0	1	1	0

change to 1

$H = B(1/2) = 1$



(a) The attribute A_4 is selected for the first (root) node of the decision tree. Calculate the information gain based on splitting on A_4.

$H(0|A_4) = \frac{1}{2} B(1/4) + \frac{1}{2} B(3/4) = 0.8113$

$\text{Gain} = 1 - 0.8113 = 0.1887$

(b) Consider the **TRUE** branch of the A_4 split. Identify whether an additional split is needed, and if so which attribute to select next.

A_1: $\frac{1}{2} B(0) + \frac{1}{2} B(\frac{1}{2}) = 0.5$ ← CHOOSE A_1

A_2: $\frac{1}{4} B(0) + \frac{3}{4} B(1/3) = 0.6887$

A_3: $\frac{3}{4} B(1/3) + \frac{1}{4} B(0) = 0.6887$

c) FALSE BRANCH

A_1: $\frac{3}{4} B(2/3) + \frac{1}{4} B(1) = 0.6887$

A_2: $\frac{1}{4} B(1) + \frac{3}{4} B(2/3) = 0.6887$

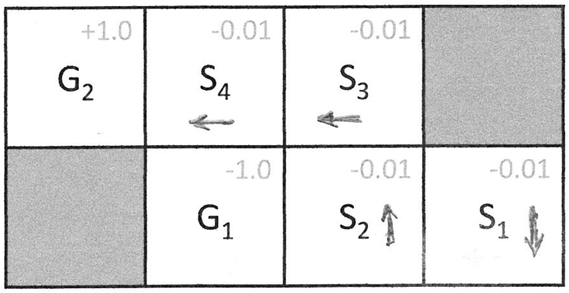
A_3: $\frac{1}{2} B(1) + \frac{1}{2} B(1/2) = 0.5$ ← choose A_3

d) A_3: $\frac{5}{8} B(3/5) + \frac{3}{8} B(1/3)$

$\rightarrow \frac{3}{4} B(2/3) + \frac{1}{4} B(0) = 0.6887$ ✓

Question 6. Markov Decision Processes

Consider the grid world shown below. The agent starts in state S_1 and is trying to reach the positive goal G_2. There is also a negative terminal G_1 which the robot wants to avoid. The reward for each of the two goals and the four states is shown in the top right of each square. Assume that there are *three* action choices in each state: *Left*, *Up*, and *Down*. Use the standard noise model for actions in which the probability of moving in the desired direction is 0.8 and there is 0.1 chance to move in each of the orthogonal directions (this is the same action model we used in class and is used in your book). Assume that rewards are not being discounted (i.e. discount factor of 1.0).



(a) Assume that the utilities for each of the states S_1 to S_4 are initialized to 0.1. Perform two rounds of value iteration, and show the utilties for each of the four states at each round.

Round 1:

$$U(S_1) = -0.01 + \max_{LUD}\left\{\overset{\leftarrow}{0.8\times(0.1)} + \overset{\uparrow}{0.1\times(0.1)} + \overset{\downarrow}{0.1\times(0.1)}\right. \overset{0.1}{} \quad \boxed{0.09} \quad \boxed{L} \text{ TIED w/ U \& D}$$

All other utilities will be same by symmetry U & D

$$U(S_2) = -0.01 + \max_{LUD}\left\{\overset{\leftarrow}{0.8\times(-1)} + \overset{\uparrow\downarrow}{0.2\times(0.1)}\right. \overset{-0.78}{} \quad L$$

$$\overset{\uparrow}{0.8\times(0.1)} + \overset{\leftarrow}{0.1\times(-1)} + \overset{\downarrow}{0.1\times(0.1)} \overset{-0.01}{} \quad \boxed{U} \text{ TIED w/ D} \quad \boxed{-0.02}$$

D will be same by symmetry

$$U(S_3) = -0.01 + \max_{LUD}\left\{\text{ALL OPTIONS SAME, IDENTICAL TO } S_1\right. \quad \boxed{0.09} \quad \boxed{L} \text{ TIED w/ U \& D}$$

$$U(S_4) = -0.01 + \max_{LUD}\left\{\overset{\leftarrow}{0.8\times(1.0)} + \overset{\uparrow}{0.1\times(0.1)} + \overset{\downarrow}{0.1\times(-1)}\right. \overset{0.71}{} \quad \boxed{L} \quad \boxed{0.7}$$

$$\overset{\uparrow\ \rightarrow}{0.9\times(0.1)} + \overset{\leftarrow}{0.1\times(1.0)} \overset{0.19}{} \quad U$$

$$\overset{\downarrow}{0.8\times(-1)} + \overset{\leftarrow}{0.1\times(1)} + \overset{\rightarrow}{0.1\times(0.1)} \overset{-0.69}{} \quad D$$

Round 2:

$$U(S_1) = -0.01 + \max_{LUD} \Big\{ \text{L IS NOT COMPETITIVE W/ NEGATNE UTIL} \quad L$$

$$\overset{\uparrow \rightarrow}{0.9\times(0.09)} + \overset{\leftarrow}{0.1\times(-0.02)} \overset{0.079}{} \quad U \quad \boxed{0.069 \ D} \quad \text{BREAK TIE ALPHABET}$$

D will be same by symmetry

$$U(S_2) = -0.01 + \max_{LUO} \Big\{ \text{L CAN'T WIN}$$

$$\overset{\uparrow}{0.8\times(0.09)} + \overset{\leftarrow}{0.1\times(-1)} + \overset{\rightarrow}{0.1\times(0.09)} \overset{-0.019}{} \quad \boxed{U} \quad \boxed{-0.029}$$

D CAN'T WAN BECAUSE 0.09 > -0.02

$$U(S_3) = -0.01 + \max_{LUD} \Big\{ \overset{\leftarrow}{0.8\times(0.7)} + \overset{\downarrow}{0.1\times(-0.02)} + \overset{\uparrow}{0.1\times(0.09)} \overset{0.567}{} \quad \boxed{L} \quad \boxed{0.557}$$

U & D CAN'T WIN

$$U(S_4) = -0.01 + \max_{LUD} \Big\{ \overset{\leftarrow}{0.8\times(1)} + \overset{\uparrow}{0.1\times(0.7)} + \overset{\downarrow}{0.1\times(-1)} \overset{0.77}{} \quad \boxed{L} \quad \boxed{0.76}$$

U & D CAN'T WIN

(b) For each state in the figure above, draw an arrow indicating the optimal policy following the 2nd iteration of value iteration.

See figure, ties broken alphabetically in S_1

(c) Now assume that the robot is moving on an infinite plane with no obstacles or goals, and the reward at each state is 0.5. Using a discounting factor of 0.9, what is the expected utility of the optimal policy?

The question is asking about infinite horizon case

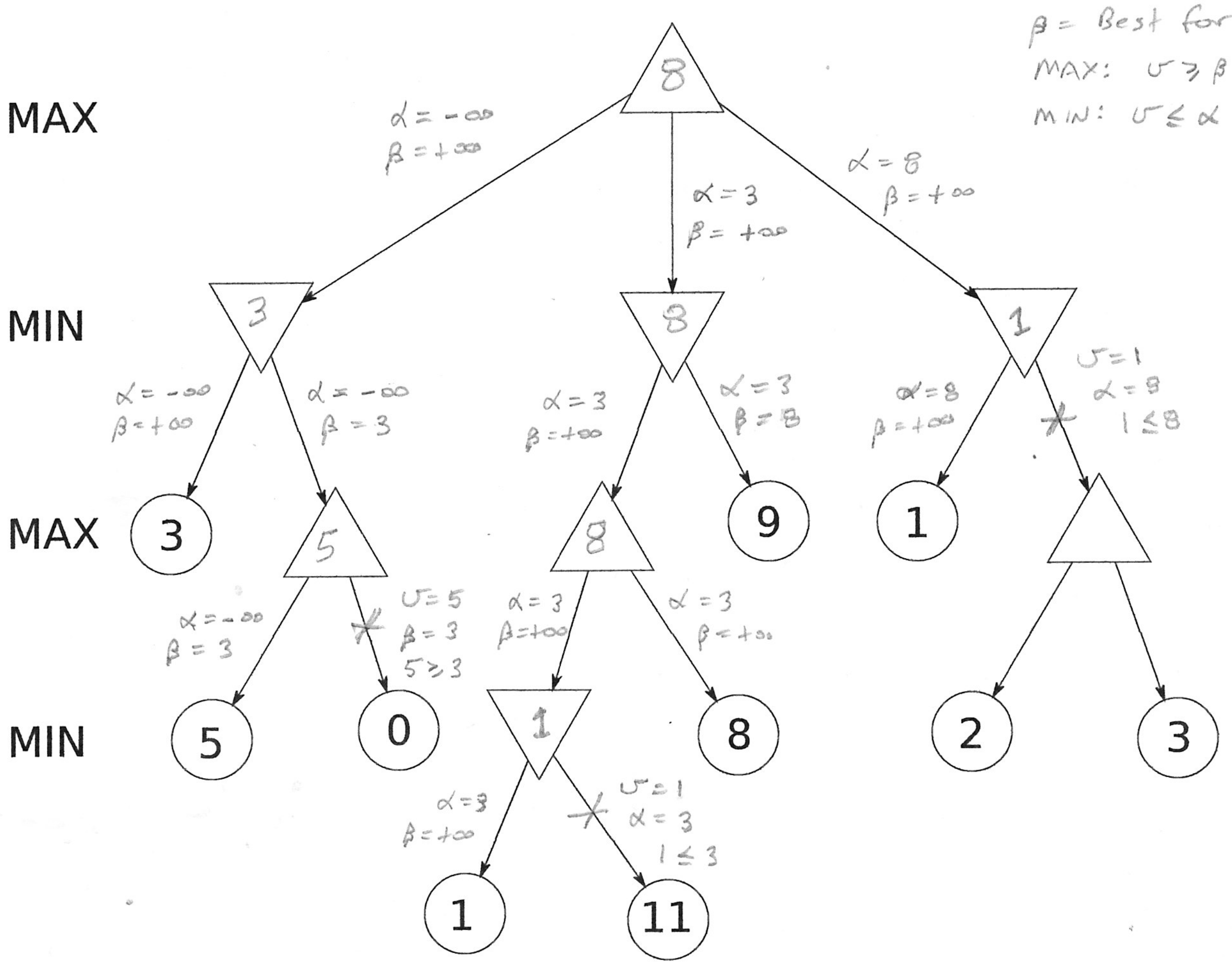
$$U = \sum_{t=0}^{\infty} \gamma^t R(S_t) = 0.5 \sum_{t=0}^{\infty} (0.9)^t = \frac{0.5}{1-0.9} = 5$$

Note: Since all directions equal, all actions equal. Then t is just how many cells visited

See (17.1) in book

Question 7. Consider the following tree for a minimax game. Note: End utilities given are always with respect to MAX, even if they don't appear on a MAX layer.

α = Best for MAX
β = Best for MIN
MAX: $v \geq \beta$
MIN: $v \leq \alpha$



Note: The α, β values on each branch are the values that are passed in to the function call for the node
α can only change across the actions for a MAX node
β can only change across the actions for a MIN node

Use the **Minimax algorithm with Alpha-Beta pruning** to identify the best strategy for MAX. Assume the algorithm always searches successors from left to right. Cross out all branches that are pruned. For each prune, write the value of v and α or β (whichever applies to the pruning decision). Fill in the value of every node, as the algorithm would determine.