Assignment 2

Probabilistic Inference

Part A Deadline: Wednesday, November 23, end of day
Part B Deadline: Wednesday, December 7, end of day
Perfect score: 100 points (extra credit available)

Assignment Instructions:

Teams: Assignments should be completed by groups of students (up to 3 - from any section). No additional credit will be given for students working individually or in groups of 2. You are encouraged to form a team of three students. Please inform the TAs as soon as possible about the members of your team so they can update the scoring spreadsheet and no later than Wednesday, November 16. Only one member of the team should submit the assignment. You can find the TAs' contact information on the course's syllabus.

For indicating the members of your team for this assignment you can use the following Google form: `https://forms.gle/7ZHeKDJgBykbbhFa6`.

Report Submission Rules: Submit your reports electronically as a PDF document through Cavnas. For your reports, do not submit Word documents, raw text, or hardcopies etc. Make sure to generate and submit a PDF instead regardless of how you generated the material. *Each team of students should submit only a single copy of their solutions and indicate all team members (name, netID and section) on their submission*.

Program Evaluation: For the programming component, you need to also submit a compressed file via Canvas, which contains your results and/or code as requested by the assignment. You will need to make sure that your program is executable by the TAs by following the instructions indicated here.

Late Submissions: No late submissions are allowed with the exception of extreme circumstances. Any delay in submitting the assignment will impact your ability to prepare for the final exam.

Start working on the assignment early!

Extra Credit for LaTeX: You will receive 6% extra credit points if you submit your answers as a typeset PDF (i.e., one that has been prepared by using LaTeX, in which case you should also submit electronically the source LaTeX code for the report). There will be a 3% bonus for electronically prepared answers (e.g., on MS Word, etc.), which are not typeset with LaTeX.Resources on how to use LaTeX are available here: `https://robotics.cs.rutgers.edu/pracsys/courses/intro-to-computational-robotics/#latex`

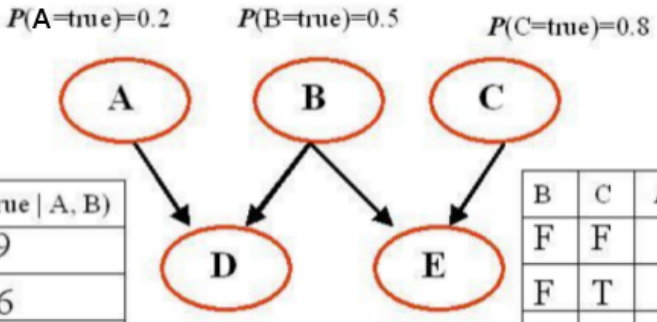
The bonuses are computed as functions of your score, i.e., if you were to receive 50 points out of 100 and you typesetted your answer with LaTeX, then you will get a score of 53. If you want to submit a handwritten report, scan it and submit a PDF but you will not receive any extra credit points. If you choose to submit PDFs of handwritten answers and we are not able to read them, you will not be awarded any points for the part of the solution that is unreadable. We will not accept hardcopy submissions.

Precision: Try to be precise in your description and thorough in your evaluation. Have in mind that you are trying to convince a very skeptical reader (and computer scientists are the worst kind...) that your answers are correct.

Collusion, Plagiarism, etc.: Each team must prepare its solutions independently from other teams, i.e., without using common code, notes or worksheets with other students or trying to solve problems in collaboration with other teams. If you are asked to implement an algorithm yourself, you should do so and not use external code. You must indicate any external sources you have used in the preparation of your solution. Do not plagiarize online sources and in general make sure you do not violate any of the academic standards of the course, the department or the university (the standards are available through the course's syllabus). Failure to follow these rules may result in failure in the course.

PART A. Due November 23.

Question 1: [10 points] Consider the following Bayesian network, where variables *A* through *E* are all Boolean valued:



A	B	P(D=true | A, B)
F	F	0.9
F	T	0.6
T	F	0.5
T	T	0.1

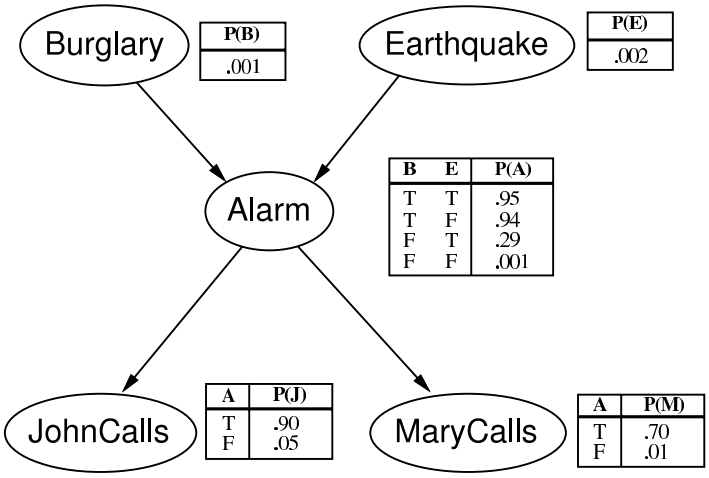
B	C	P(E=true | B, C)
F	F	0.2
F	T	0.4
T	F	0.8
T	T	0.3

a) What is the probability that all five of these Boolean variables are simultaneously true?
[Hint: You have to compute the joint probability distribution (JPD). The structure of the Bayesian network suggests how the JPD is decomposed to the conditional probabilities available.]

b) What is the probability that all five of these Boolean variables are simultaneously false?
[Hint: Answer similarly to above.]

c) What is the probability that A is false given that the four other variables are all known to be true?
[Hint: Try to use the definition of the conditional probability and the structure of the network. For probabilities that can not be computed directly from the network, remember the following normalization trick: if $P(x) = \alpha \cdot f(x)$ and $P(\neg x) = \alpha \cdot f(\neg x)$, then you can compute the normalization factor as $\alpha = \frac{1.0}{f(x)+f(\neg x)}$, since $P(x) + P(\neg x) = 1.0$.]

Question 2: [10 points] For this problem, check the Variable Elimination algorithm in your book. Also consider the Bayesian network from the "burglary" example.



a) Apply variable elimination to the query:

$$P(Burglary|JohnsCalls = true, MaryCalls = true)$$

and show in detail the calculations that take place. Use your book to confirm that your answer is correct.

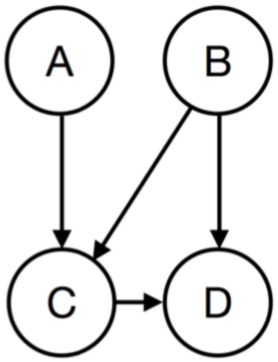
b) Count the number of arithmetic operations performed (additions, multiplications, divisions), and compare it against the number of operations performed by the tree enumeration algorithm.

c) Suppose a Bayesian network has the from of a *chain*: a sequence of Boolean variables $X_1, \ldots X_n$ where $Parents(X_i) = \{X_{i-1}\}$ for $i = 2, \ldots, n$. What is the complexity of computing $P(X_1|X_n = true)$ using enumeration? What is the complexity with variable elimination?

Question 3: [15 points] In this problem, you will implement two sampling techniques (rejection sampling and likelihood weighting) to perform approximate inference on the following Bayesian network.

A	$P(A)$
a	0.0

B	$P(B)$
b	0.9



A	B	$P(c \mid A, B)$
a	b	0.2
a	$\neg b$	0.6
$\neg a$	b	0.5
$\neg a$	$\neg b$	0.0

B	C	$P(d \mid B, C)$
b	c	0.75
b	$\neg c$	0.1
$\neg b$	c	0.5
$\neg b$	$\neg c$	0.2

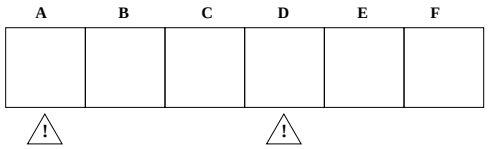
a) Compute the probabilities of $P(d|c)$, $P(b|c)$, and $P(d|\neg a, b)$ by enumeration.

b) Now employ rejection sampling and likelihood weighting to approximate the same three conditional probabilities. Use 1000 samples, and document your results. How do the approximations and the actual values compare?

c) Next focus on $P(d|c)$. We know that the accuracy of the approximation depends on the number of samples used. For each of the sampling methods, plot the probability as a function of the number of samples used. Do you notice a large divergence in the convergence rates among the two methods?

d) Construct a query on this Bayesian Network such that the convergence and effectiveness of rejection sampling is noticeably worse than that of likelihood weighting. List the query you are using, and provide the probability plot as a function of samples used. Why is it the case that rejection sampling is noticeably worse for this query?

Question 4: [15 points] You are an interplanetary search and rescue expert who has just received an urgent message: a rover on Mercury has fallen and become trapped in Death Ravine, a deep, narrow gorge on the borders of enemy territory. You zoom over to Mercury to investigate the situation. Death Ravine is a narrow gorge 6 miles long, as shown below. There are volcanic vents at locations A and D, indicated by the triangular danger symbols at those locations.



The rover was heavily damaged in the fall, and as a result, most of its sensors are broken. The only ones still functioning are its thermometers, which register only two levels: *hot* and *cold*. The rover sends back evidence $E = hot$ when it is at a volcanic vent (A and D), and $E = cold$ otherwise. There is no chance of a mistaken reading. The rover fell into the gorge at position A on day 1, so $X_1 = A$. Let the rover's position on day t be X_t, where $X_t \in \{A, B, C, D, E, F\}$ on any day. The rover is still executing its original programming, trying to move 1 mile east (i.e., right, towards F) every day. Because of the damage, however, it only moves east with probability 0.80 on any one day, and it stays in place with probability 0.20 on any one day. Your job is to figure out where the rover is, so that you can dispatch your rescue-bot.

a) Filtering: Three days have passed since the rover fell into the ravine. The observations were $(E_1 = hot, E_2 = cold, E_3 = cold)$. What is $P(X_3 \mid hot_1, cold_2, cold_3)$, the probability distribution over the rover's position on day 3, given the observations? (This is a probability distribution over the six possible positions).

b) Smoothing: What is $P(X_2 \mid hot_1, cold_2, cold_3)$, i.e., the probability distribution over the rover's position on day 2, given the observations for the first three days? (Again, provide a probability distribution over all six possible positions).

c) Most Likely Explanation: What is the most likely sequence of the rover's positions in the three days given the observations $(E_1 = hot, E_2 = cold, E_3 = cold)$?

d) Prediction: What is $P(hot_4, hot_5, cold_6 \mid hot_1, cold_2, cold_3)$? i.e., the probability of observing the sequence hot_4 and hot_5 and $cold_6$ in days 4,5,6 respectively, given the previous observations in days 1,2, and 3? (This is a single value, not a distribution).

e) Prediction: You decide to attempt to rescue the rover on day 4. However, the transmission of E_4 seems to have been corrupted, and so it is not observed. What is the probability distribution for the rover's position 4 given the same evidence, $P(X_4 \mid hot_1, cold_2, cold_3)$?
The same thing happens again on day 5. What is the probability distribution for the rover's position for day 5 given the same evidence, $P(X_5 \mid hot_1, cold_2, cold_3)$?

Question 5: [35 points: Step is 5 - Step B is 10 - Step C is 20]

H	H	T
N	N	N
N	B	H

Consider that you are placed in an unknown unblocked cell of the above 3×3 map, i.e., initially there is a probability $P(x_0) = \frac{1}{8}$ to be located in any of the unblocked cells. We denote as $(1, 2)$ the coordinates of the top middle cell, and as $(2, 3)$ the coordinates of the rightmost middle cell, where:

- N is a normal cell;
- H is a highway cell;
- T is a hard to traverse cell;
- B is a blocked cell.

You are able to move inside this world by executing actions $\alpha = \{Up, Left, Down, Right\}$. You are also equipped with a sensor that informs you about the terrain type that you are occupying after every time you are trying to move inside this world. Your objective is to figure out your location inside this world given that you had no idea initially where you were located.

Lets denote as $P(x_i|x_{i-1}, \alpha)$ the transition model for moving from cell x_{i-1} at step $(i-1)$ to cell x_i at step i given an action α. If given your true location x_{i-1} and action α, you would hit the boundary of this grid world or a blocked cell, then you stay in the same cell, i.e., $x_i = x_{i-1}$. The model is probabilistic because our motions are not executed perfectly inside this grid world. In particular, 90% of the time the action is executed correctly but 10% the action fails and the agent stays on the same cell. So, for instance if you execute action $\{Up\}$ from cell $(2, 2)$, with 90% probability you move to cell $(1, 2)$ and with 10% probability you stay at cell $(2, 2)$. If you are at cell $(3, 1)$ and move $Right$, then with 100% probability you stay at the same cell (because cell $(3, 2)$ is a blocked cell).

Furthermore, denote as $P(e_i|x_i)$ the observation model for detecting terrain type, where e_i is the observed terrain type for cell x_i. The terrain sensor is correct 90% of the time but with probability 5% it can return either of the other two terrain types (the sensor never returns "blocked cell" as you never occupy one). For instance, if you are located at cell $(2, 2)$, your terrain sensor returns with probability 90% the value N for "normal cell", with 5% probability it returns the value H for "highway cell" and with 5% probability it returns the value T for "hard to traverse cell".

*S*tep A: You are asked to compute the probability of where you are inside this grid world *after* you execute actions $\{Right, Right, Down, Down\}$ and the corresponding sensing readings are $\{N, N, H, H\}$ (you sense after you move). You are encouraged to do this both by hand and via software so as to have the capability to debug your program. Submit in your report the probabilities of where you are inside the world after each action/sensor reading pair assuming that in the beginning you have no knowledge of where you are located. This means that you need to provide four 3*x*3 maps with sets of eight probabilities indicating where you are inside this grid world after each action/sensing pair.

*S*tep B: The next objective is to scale up the size of similar filtering problems that you can solve. For this question, you will use large maps (e.g., 100 by 50, similar to Assignment 1 - feel free to reuse your visualization solution from that Assignment), where you randomly assign terrain types to each cell out of the four possible types (for each cell sample their type according to the following distribution: 50% normal cell, 20% highway cell, 20% hard to traverse cell and 10% blocked cell). First, generate "ground truth" data, i.e., generate sequences of actions and sensor readings to test your algorithm, given the above specified transition and observation models.

For this purpose, first randomly select a non-blocked cell as the initial location of your agent in the world x_o. Then, randomly generate a sequence of 100 actions, i.e., randomly select actions

from the set $\alpha = \{Up, Left, Down, Right\}$ and generate a string of length 100. For each action starting from the initial location x_0 apply:

- the transition model, i.e., 90% follow the action - when you collide stay in place, 10% stay in place), in order to get the next ground truth state x_{i+1};
- the observation model, i.e., 90% the correct terrain type and 5% one of the other two types), in order to get the "ground truth" sensor reading e_{i+1}.

Once you have generated the 100 actions, the 100 ground truth states and the 100 observations, store them in a file. Generate 10 such files per map and for 10 different maps of the world (i.e., you will generate 100 total ground truth files), as different experiments inside the large maps. The format for the file can be as follows:
x_0 y_0 :coordinates of initial point
x_i y_i :100 coordinates of the consecutive points that the agent actually visits (the ground truth states) separated by a new line
α_i :100 characters indicating the type of action executed $\{U, L, D, R\}$ separated by a new line
e_i :100 characters indicating the sensor reading $\{N, H, T\}$ collected by the agent as it moves

In your report, provide examples of ground truth paths and the corresponding action and sensor readings generated. In an accompanying compressed file, provide the corresponding output files.

PART B. Due December 7.

*S*tep C of Question 5: Demonstrate the capability of estimating the position of the agent inside the world given only the actions and observations indicated in these files. In particular, your program should be able to load a "ground truth" file and a map. Then, after each action/sensor reading pair, it should be able to compute the probability that the agent is in each cell of the map by applying the filtering algorithm. Assume that you do not know where your agent is located initially. So the prior distribution is uniform over all unblocked cells.

Visualize the different probabilities on your map (e.g., think of it as a heatmap, where you paint a cell with brighter intensity if you have a positive probability of being in the corresponding cell) or provide a capability to indicate the probability on any cell of the map (e.g., you could use your terminal to input the coordinates of a cell and then return its probability). Visualize the ground truth location of the agent inside the world at the corresponding step. Your visualization should be updated with each new action/sensor reading pair until you consume all 100 readings. Make sure that you can control the speed of your simulation so as to be able to observe the motion of the agent and the probability distribution you are computing. Use your visualization as a debugging tool for your software (e.g., you should not have positive probabilities in locations that you have no indication that your agent is occupying).

Attach example heat maps in your report after 10, 50 and 100 iterations. Indicate the ground truth path up to this point in each case.

For each of the 100 experiments, compute the error (as distance inside the grid world) between the true location of the agent and the maximum likelihood estimation (i.e., the cell with the highest probability according to the filtering algorithm) as the number of readings increases. For the computation of the maximum likelihood estimation, ties can be broken randomly. Generate a plot of the average error over all 100 experiments as the number of readings increases. For this plot, you can ignore the first 5 iterations as many cells will have the same probability in the beginning.

For each of the 100 experiments, keep track of the probability that the cell where the agent is actually located (which changes over time) is assigned by the filtering algorithm. Generate a plot of the average probability of the ground truth cell over 100 experiments as the number of readings increases. The cell will start with the uniform probability assigned to all unblocked cells.

You *may* face computational challenges in the straightforward implementation of the filtering algorithm given the sizes of the map (depending on the efficiency of your implementation and your data structure choices). If you find this to be the case, you are allowed to provide results for smaller maps (e.g., try first 50x50 maps, then 20x20). You will not be awarded all of the points but you can still collect the majority of the available points for a correct implementation for smaller maps.

Question 6: [10 points] Assume you are interested in buying a used vehicle C_1. You are also considering of taking it to a qualified mechanic and then decide whether to buy it or not. The cost of taking it to the mechanic is \$100. C_1 can be in good shape (quality q^+) or bad one (quality q^-). The mechanic might help to indicate what shape the vehicle is in. C_1 costs \$3,000 to buy and its market value is \$4,000 if in good shape; if not, \$1,400 in repairs will be needed to make it in good shape. Your estimate is that C_1 has a 70% chance of being in good shape. Assume that the utility function depends linearly on the vehicle's monetary value.

a. Calculate the expected net gain from buying C_1, given no test.

b. We also have the following information about whether the vehicle will pass the mechanic's test:

$$P(pass(c_1)|q^+(c_1)) = 0.8$$

$$P(pass(c_1)|q^-(c_1)) = 0.35$$

Use Bayes' theorem to calculate the probability that the car will pass/fail the test and hence the probability that it is in good/ bad shape given what the mechanic will tell you.
[Hint: Compute the four probabilities: $P(q^+|Pass)$, $P(q^-|Pass)$, $P(q^+|\neg Pass)$, $P(q^-|\neg Pass)$]

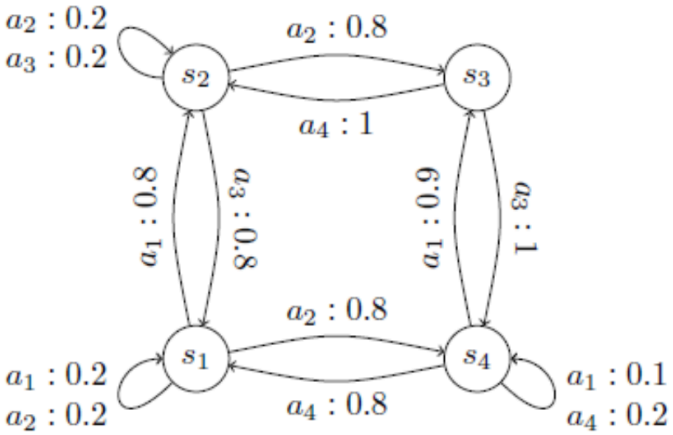
c. What is the best decision given either a pass or a fail? What is the expected utility in each case?
[Hint: Use the probabilities from the previous question.]

d. What is the value of optimal information for the mechanic's test? Will you take C_1 to the mechanic or not?
[Hint: You can easily answer this based on the answers from questions a) and c).]

Question 7: [15 points] Consider the specification of a Markov Decision Process according to the following figure. Code your own implementation of Value Iteration and compute the optimal policy as well as the optimum utilities for this challenge.

Indicate the original utilities you used in order to start the process. Provide at least 5 intermediate results (in terms of optimum utilities and policies) depending on the number of iterations needed for convergence as well as the final results. Describe your implementation and your convergence criterion. Report computation time and number of iterations.

s	$r(s)$
s_1	0
s_2	0
s_3	1
s_4	0



s_i	a	s_j	$T(s_i, a, s_j)$
s_1	a_1	s_1	0.2
s_1	a_1	s_2	0.8
s_1	a_2	s_1	0.2
s_1	a_2	s_4	0.8
s_2	a_2	s_2	0.2
s_2	a_2	s_3	0.8
s_2	a_3	s_2	0.2
s_2	a_3	s_1	0.8
s_3	a_4	s_2	1
s_3	a_3	s_4	1
s_4	a_1	s_4	0.1
s_4	a_1	s_3	0.9
s_4	a_4	s_4	0.2
s_4	a_4	s_1	0.8