Assignment 3

Bayesian Inference, Temporal State Estimation and Decision Making under Uncertainty

Deadline: April 11, 11:59pm
Perfect score: 110 points.

Assignment Instructions:

Teams: Assignments should be completed by pairs of students. No additional credit will be given for students working individually. You are strongly encouraged to form a team of two students. If you have not done so already, please inform the TAs as soon as possible about the members of your team so they can update the scoring spreadsheet (find the TAs' contact info under the course's website: `http://www.pracsyslab.org/cs440`).

Submission Rules: Submit your reports electronically as a PDF document through Sakai (`sakai.rutgers.edu`). For programming questions, you need to also submit a compressed file via Sakai, which contains your code. Do not submit Word documents, raw text, or hardcopies etc. Make sure to generate and submit a PDF instead. Each team of students should submit only a single copy of their solutions and indicate all team members on their submission. Failure to follow these rules will result in lower grade in the assignment.

Program Demonstrations: You will need to demonstrate your program to the TAs on a date after the deadline. The schedule of the demonstrations will be coordinated by the TAs. During the demonstration you have to use the file submitted on Sakai and execute it either on your laptop computer (easier) or an available machine at CBIM, where the TAs are located (you probably need to coordinate this ahead of time). You will also be asked to describe the architecture of your implementation and key algorithmic aspects of the project. You need to make sure that you are able to complete the demonstration and answer the TAs' questions within the allotted 12 minutes of time for each team. If your program is not directly running on the computer you are using and you have to spend time to configure your computer, this counts against your allotted time.

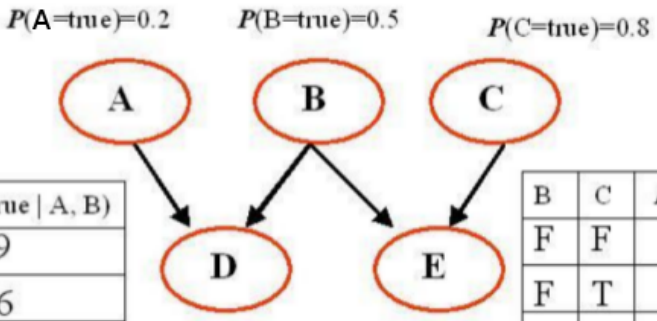
Late Submissions: No late submissions are allowed. You will be awarded 0 points for late assignments!

Extra Credit for LaTeX: You will receive 10% extra credit points if you submit your answers as a typeset PDF (using LaTeX, in which case you should also submit electronically your source code). Resources on how to use LaTeX are available on the course's website (`http://www.pracsyslab.org/cs440`). There will be a 5% bonus for electronically prepared answers (e.g., on MS Word, etc.) that are not typeset. If you want to submit a handwritten report, scan it and submit a PDF via Sakai. We will not accept hardcopies. If you choose to submit 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.

Precision: Try to be precise. 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. 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 website: `http://www.pracsyslab.org/cs440`). Failure to follow these rules may result in failure in the course.

Question 1: 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 $P(x) + P(\neg x) = 1.0$.]

[15 points]

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

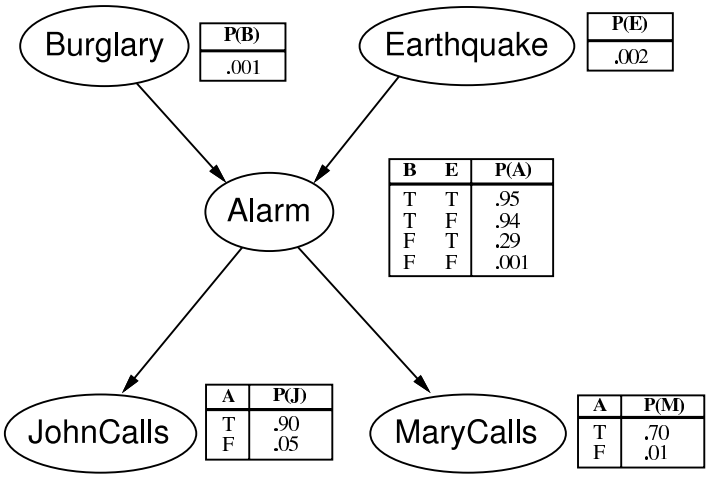
a) Apply variable elimination to the query:

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

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?



[15 points]

Question 3: One method for approximate inference in Bayesian Networks is the Markov Chain Monte Carlo (MCMC) approach. This method depends on the important property that a variable in a Bayesian network is independent from any other variable in the network given its Markov Blanket.

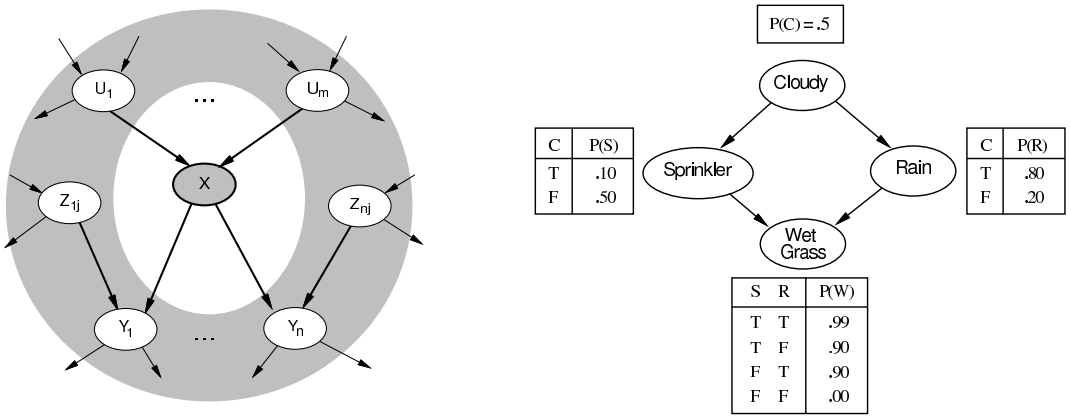


Figure 1: (left) The Markov Blanket of variable X (right) The Rain/Sprinkler network.

a) Prove that

$$P(X|MB(X)) = \alpha\, P(X|U_1, \ldots, U_m) \prod_{Y_i} P(Y_i|Z_{i1} \ldots)$$

where $MB(X)$ is the Markov Blanket of variable X.

b) Consider the query

$$P(Rain|Sprinkler = true, WetGrass = true)$$

in the Rain/Sprinkler network and how MCMC would answer it. How many possible states are there for the approach to consider given the network and the available evidence variables?

c) Calculate the transition matrix Q that stores the probabilities $P(y \rightarrow y')$ for all the states y, y'. If the Markov Chain has n states, then the transition matrix has size $n \times n$ and you should compute n^2 probabilities.

[Hint: Entries on the diagonal of the matrix correspond to self-loops, i.e., remaining in the same state. Such transitions can occur by sampling either variable. Entries where one variable is different between y and y', must sample that one variable. Entries where two or more variables change cannot occur, since in MCMC only one variable is allowed to change at each transition.]

[15 points]

Question 4: 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).]

[15 points]

5. Programming Component

[50 points]

Small Scale Example

H	H	T
N	N	N
N	B	H

Consider that you are placed in an unknown 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 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 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".

Question A: [5 points] 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 by hand and through a program so as to have the capability to debug your solution. 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 3x3 maps with eight probabilities indicating where you are inside this grid world after each action/sensing pair.

The above challenge corresponds to the filtering problem. A description of the filtering algorithm can be found in the lecture notes and the textbook [RNC+09].

Question B: [5 points] Another way to formulate the problem is to try to compute the most likely sequence of states that you have gone through given the four actions and sensing readings. Please provide the "most likely sequence" for the above problem by applying the Viterbi algorithm. You are encouraged again to compute the solution by hand and via a program in order to validate the

accuracy of your implementation. You need to indicate the most likely sequence finishing at each state and its associated probability after each action/sensing reading (these are the intermediate steps of the Viterbi algorithm), as well as the most likely sequence overall after all four readings.

Generating Ground Truth Data

Question C: [5 points] The next objective is to scale up the type of filtering and "most likely explanation" problems that you can solve. For this question, you will use large maps, where you randomly assign terrain types to each cell out of the four possible types (40% normal cells, 20% highway cells, 20% hard to traverse and 10% blocked cells). First, generate "ground truth" data, i.e., generate sequences of actions and sensor readings to test your algorithm.

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 for 10 different maps of the world (i.e., 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 goes through 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.

Filtering and Viterbi Algorithms in Large Maps

Question D: [8 points] Show 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 compute the probability that the agent is in each cell of the map by applying the filtering algorithm. Visualize the different probabilities on your map (e.g., think of a heatmap) or provide a capability to indicate the probability on any cell of the map. 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.

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.

Question E: [5 points] 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. Here you can start with the uniform probability assigned to the cell in the beginning of this process.

Question F: [7 points] Execute the Viterbi algorithm over the corresponding data in order to estimate the most likely trajectory followed by the algorithm. Your program should incrementally compute the most likely trajectories that finish on different cells according to the Viterbi algorithm as the action/sensor readings arrive. Visualize the 10 most likely trajectories that end up in different cells inside the world after each action/sensor reading pair. Make sure to differentiate and clearly indicate the most likely trajectory.

Provide in your report example visualizations generated by your software of these 10 most likely trajectories after 10, 50 and 100 iterations. If it becomes computationally expensive to get up to 100 iterations, provide the results up to the point that you were able to run the computation.

Question G: [5 points] For each of the 100 experiments, compute the error between the most likely sequence computed after 100 iterations and the true trajectory. The error is the distance between the true location of the agent and the estimated location by the Viterbi algorithm for the corresponding time step. Provide a plot for the average error over all experiments.

Scalability Note: You *may* face computational challenges in the straightforward implementation of the above algorithms given the sizes of the map. If you find this to be the case, you are allowed to provide results for smaller maps (e.g., try first 20x20 maps, then 50x50) at the expense of a grade penalty (20% and 10% for the corresponding sizes indicated here - for Questions D,E,F,G).

Computational Approximations

Question H: [10 points] Identify ways of how you can provide solutions for these problems that can scale to larger maps, where it becomes difficult to keep track of all the probabilities in an efficient manner. For instance, you can consider approximation ideas based on sampling and pruning the space over which you keep tracking the corresponding probabilities.

Implement this idea to the largest size of a map that you can address without performing any approximation so as to reduce the computational overhead of your solution. Demonstrate the computational speedup achieved by your approximation idea by reporting computation times for updating the probabilities of the cells you are keeping track of every time you get a new action/sensor reading pair.

Report, however, the error that the approximation introduces in the computation of the filtering/most likely sequence probabilities. In particular, in the case of filtering, report the difference in the probability computed by the filtering algorithm for the "ground truth" cell with and without the approximation. In the case of the Viterbi algorithm, report the difference between the most likely sequence computed with and without your approximation (as distance between cells at the corresponding time step).

Scalability Note: Show how large are the maps that you can deal with given your approximation approach. If with your approximation you can now deal with the 100x100 maps, then you will not get any grade penalty if you did not manage to run the exhaustive computations in the full maps.

References

[RNC+09] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach 3rd Edition*, volume 2. Prentice hall Upper Saddle River, 2009.