15-780: Graduate AI

Homework Assignment #3 Solutions

Out: March 23, 2015
Due: April 6, 2015 5 PM

Collaboration Policy: You may discuss the problems with others, but you must write all code and your writeup independently.

Turning In: Please email your assignment by the due date to shayand@cs.cmu.edu and vdperera@cs.cmu.edu. Your solutions should be submitted as a **single** pdf file. If your solutions are handwritten, **scan** them and make sure they are legible and clear. Please submit your code in separate files and provide instructions on how to run it.

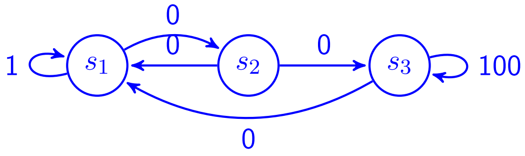
1 R-MAX

R-MAX makes the assumption that we know the maximum possible reward. In the real world, this might not always be the case. Let's explore some possible modifications to the algorithm, when we don't know the reward.

(a) Suppose we have an upper bound for the reward (which we are confident is actually an upper bound) and we set R_{max} to be this upper bound. When this bound is very loose, how will the algorithm behave?

If the bound is very loose (i.e. the upper bound is much larger than the actual maximum possible reward), then the algorithm will visit every (state, action) pair the threshold number of times since any unknown (state, action) pair will appear to give a much higher reward than known (state, action) pairs.

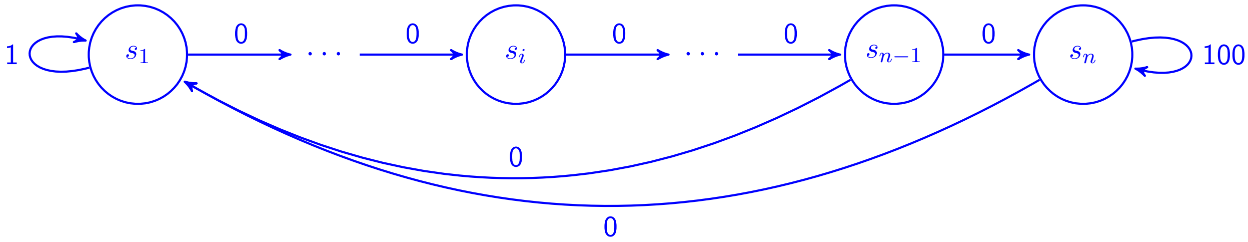
(b) Now suppose we initialize $R_{max} = 0$ (which we assume is the minimum possible reward), and every time a (state, action) pair becomes known with a reward $\rho > R_{max}$, we set $R_{max} = \rho$ (and modify all our unknown states accordingly). Intuitively this might seem like it should work, but in some cases it does not. Give a brief description of a simple MDP where this fails (i.e. where we will never find the optimal policy).



Suppose we start in state s_1 in the deterministic MDP shown above. Notice that if at any point we are in s_1 and both actions in s_1 are known, but the action that stays in s_3 is not known, then the algorithm will assume $R_{max} = 1$, and hence will stay in s_1, since it thinks it can achieve R_{max} at every time step by doing so, while transitioning to s_2 will give it a reward of 0 for one time step. This means it won't find the optimal policy of staying in s_3 at every step. We will show that this situation will *always* occur.

Let N be the number of times we have visited s_2 so far in the algorithm and suppose the threshold is $m > 1$. Then we know that the number of times we have visited s_3 is at most $\lceil N \rceil / 2$, because we can only enter s_3 from s_2, and due to balanced wandering that will happen half the time we are in s_2. We also know that the number of times we have taken the action that transitions from s_1 to s_2 at least N times since we can only enter s_2 from s_1. Now suppose $N = m$. Then we know that the action that goes from s_1 to s_2 is known, and at least by the next time we are in s_1 the self-loop in s_1 will be known. We also know that at that point the self-loop in s_3 cannot be known, since we haven't visited s_3 m times yet. Thus we will stay in s_1 for the rest of time.

(c) Now suppose we use the same algorithm as in part (b), but instead we replace $R_{max} = \rho + \delta$ for some δ (which may depend on the discount factor γ). Either give a brief explanation of why this won't fail like in part (b), or show that for any value of δ, you can construct an MDP where this fails.



The MDP above is identical to the one in part (b) except that s_2 is now labeled s_{n-1} and there are $n - 3$ additional states that form a chain between s_1 and s_{n-1} (i.e. there is only one available action in each of those states). Notice that since all the actions are enforced between s_1 and s_{n-1} the behavior of the algorithm in part (b) is the same in this MDP as in the previous MDP except that all of the actions along the chain become known as well. Further notice that as long as $\gamma^{n-2}(1 + \delta) < 1$, we would prefer to stay in s_1 than make the actions in s_n known. Thus, for each δ we can construct an MDP by setting

$$n = \log_\gamma \left(\frac{1}{1 + \delta} \right) + 2$$

2 Value of Information

Consider a modified version of the example given in class. An oil company wants to buy the rights to drill in one of five blocks of land. It knows that exactly one block contains \$10 million worth of oil and all the other blocks contain \$0 worth of oil. You can take one or both of the following actions:

- Pay a seismologist \$2.5 million to survey one block of land and tell you with certainty whether it is oil-rich or not.
- Pay a fortune teller \$1 million to tell you with certainty that the oil-rich block of land is one of two blocks.

Answer the following, justifying your answers with value of information calculations. For simplicity, you may write \$1 to mean \$1 million.

(a) Consider the myopic algorithm where you choose the action whose value of information minus cost is the greatest and repeat until no actions are left or all actions give negative reward. What is the policy if the company uses this algorithm and what is the expected payoff?

Let S be the action of using the seismologist's service and F be the action of using the fortune teller's service. If we use the seismologist, with probability $\frac{4}{5}$ she will not identify the oil-rich block, which means we will choose randomly from among the four remaining, and with probability $\frac{1}{5}$ she will identify the oil-rich block, which means you get that block with certainty. Thus:

$$VoI(S) = \frac{4}{5}(\frac{1}{4}\$8) + \frac{1}{5}(\$8) = \$2$$

Since we have to pay the seismologist \$2.5, we will not take this action! If we use the fortune teller, he will identify that the oil-rich block is one of two blocks, and we will choose randomly from among those. Thus:

$$VoI(F) = \frac{1}{2}(\$8) = \$4$$

Since we only have to pay the fortune teller \$1, we will take this action. Now notice that since only two blocks remain after using the fortune teller, the seismologist will identify the oil-rich block with certainty, but recall that we already paid the fortune teller \$1. Thus:

$$VoI(S|F) = \$8 - \$1 = \$7$$

Since we have to pay the seismologist only \$2.5, we will take this action. Thus the myopic policy is to take S then F and then purchase the unique block of land that was identified as being oil-rich. The expected payoff is \$4.5.

(b) What is the optimal policy and expected payoff?

Notice that the only policy not considered above is S then F, but in that case we won't identify the block of land with certainty, but we will pay the same amount as the myopic policy, so doing F then S is better! Thus the optimal policy is the same as the myopic policy and hence has the same payoff.

(c) Now suppose the fortune teller rises his cost to \$4.5 million dollars. How do your answers to part (a) and (b) change?

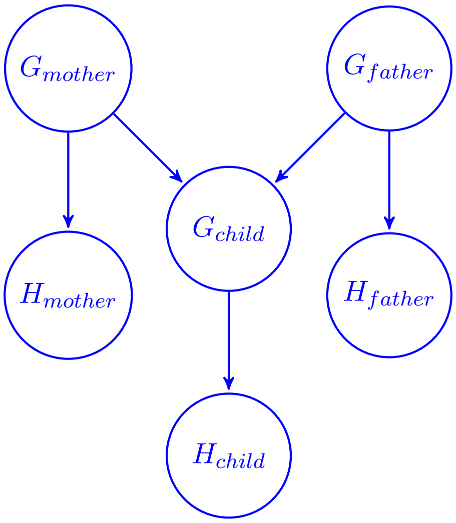
In this case we would not take F in the myopic policy, since the cost is more than the value of information, and since we also wouldn't take S, we will just choose to purchase a block of randomly, with an expected payoff of \$0. The optimal policy is still the same as before, with a payoff of $\$8 - \$2.5 - \$4.5 = \$1$. Thus, being myopic hurts us in this case!

3 Bayes Nets

Let H_i be a random variable taking on values l or r that denotes the handedness of some individual i. A simple hypothesis for handedness is that it is inherited in the following way:

- there is a single gene G_i that effects H_i,
- $H_i = G_i$ with probability $p > 0.5$,
- the gene is inherited from a single parent, and it is equally likely to be inherited from either,
- there is a small non-zero probability m that the gene mutates after inheritance (e.g. if the child inherits from the father and $G_{father} = r$, then with probability m, $G_{child} = l$).

(a) Draw a Bayes' network with nodes G_i and H_i for $i \in \{child, mother, father\}$ that shows this hypothesis.



(b) Answer the following using d-separation arguments.

(a) is H_{mother} independent of H_{father}?

(b) is H_{mother} independent of H_{father} given H_{child}?

(c) is H_{child} independent of G_{mother} given G_{child}?

(d) is H_{child} independent of G_{mother} given H_{mother}?

Notice that in this graph there is only path from any node to any other node. Thus, to see if two nodes are independent, we only have to identify if that path is blocked or not.

(a) Yes, because on the path of interest, there are head-to-head edges going into G_{child}, which is not in the evidence set and neither is its descendent H_{child}.

(b) No, because on the path of interest, there are head-to-head edges going into G_{child}, and its descendent H_{child} is in the evidence set.

(c) Yes, because on the path of interest there are head-to-tail edges going into G_{child}, which is in the evidence set.

(d) No, because on the path of interest, there are no head-to-head edges and there are no nodes in the evidence set.

(c) Give the conditional probability table for G_{child}.

G_{mother}	G_{father}	$P(G_{child} = l \| \cdots)$	$P(G_{child} = r \| \cdots)$
l	l	$1 - m$	m
l	r	0.5	0.5
r	l	0.5	0.5
r	r	m	$1 - m$

(d) Suppose $P(G_{father} = l) = P(G_{mother} = l) = x$. Derive an expression for $P(G_{child} = l)$ in terms of x and m by conditioning on the parent nodes.

$$\begin{aligned} P(G_{child} = l) &= \sum_{g_m, g_f} P(G_{child} = l | G_{mother} = g_m, G_{father} = g_f) P(G_{mother} = g_m, G_{father} = g_f) \\ &= \sum_{g_m, g_f} P(G_{child} = l | G_{mother} = g_m, G_{father} = g_f) P(G_{mother} = g_m) P(G_{father} = g_f) \\ &= (1 - m)x^2 + 0.5x(1 - x) + 0.5(1 - x)x + m(1 - x)^2 = x + m - 2mx \end{aligned}$$

(e) Suppose genetic equilibrium holds, i.e. that the distribution of genes in every generation is the same. Calculate x. Do you think the hypothesis for handedness described in this question holds? Explain.

$$P(G_{child} = l) = P(G_{father} = l) \implies x + m - 2mx = x \implies 2mx = m \implies x = 0.5$$

$P(G_{child} = l) = 0.5$ implies $P(H_{child} = l) = 0.5p + 0.5(1 - p) = 0.5$, which means half of the population should be left-handed under genetic equilibrium. Since we know only a small fraction of the population is left-handed, the hypothesis for handedness described in this question must be incorrect (if we assume genetic equilibrium).

4 Bayesian Knowledge Tracing

The Bayesian Knowledge Tracing (BKT) model (Corbett and Andersen, 1994) is a Bayesian network used to model students' mastery of skills when interacting with an intelligent tutoring system (ITS) or educational game for example. The BKT model is simply an HMM with one binary-valued state (whether the student has mastered the skill or not), and one binary-valued observation (whether the student answers a question correct or not). The student is repeatedly given problems that should help them learn a skill, and the BKT model is used to assign a probability to how likely the student is to have mastered the skill and to predict the probability of the student answering questions correctly in the future. The analogous HMM is shown in Figure 1. K_i is the students internal state after i problems (i.e. it is 1 if the student has mastered the skill and 0 if the student has not), and C_i represents the students answer to the ith problem (i.e. it is 1 if it was correct, and 0 if it was incorrect).

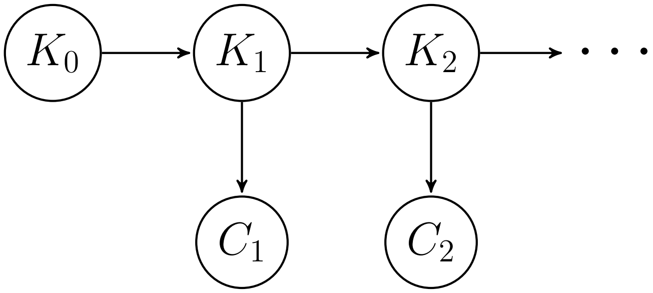


Figure 1: HMM for the BKT model

The standard BKT model assumes a student never forgets a skill, e.g. $P(K_i = 0|K_{i-1} = 1) = 0$. Therefore the four possible parameters of the BKT model are described in Table 1.

Parameter	Formula	Interpretation
L_0	$P(K_0 = 1)$	The initial probability that a student has mastered the skill
T	$P(K_i = 1\|K_{i-1} = 0)$	The probability that the student transitions from not knowing the skill to knowing the skill
G	$P(C_i = 1\|K_i = 0)$	The probability of **guessing**, answering a question correctly when the skill is not mastered
S	$P(C_i = 0\|K_i = 1)$	The probability of **slipping**, answering a question incorrectly, despite the skill being mastered

Table 1: BKT Model Parameter Summary

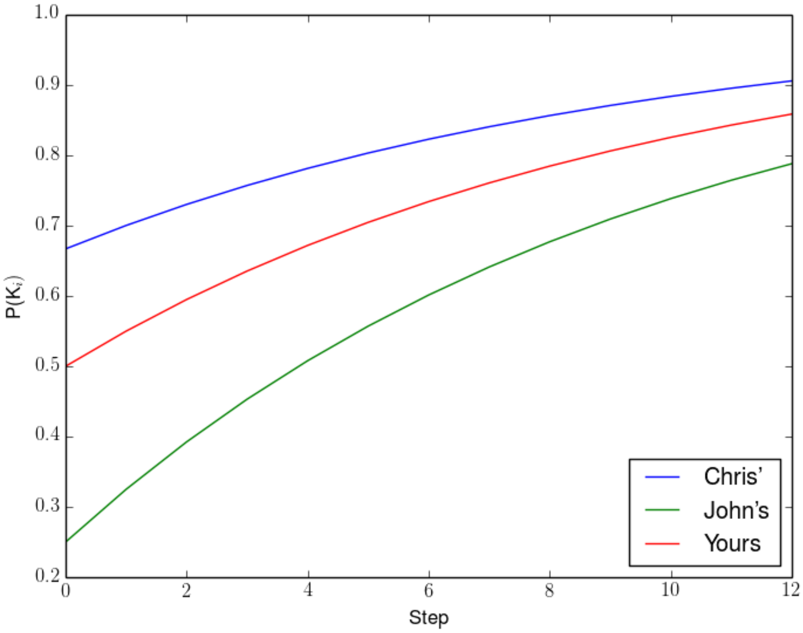
Chris and his friend John both come up with models of how students learn how to solve value of information problems, and they get into an argument over which of their models is better. Chris thinks that students can never guess on a question if they haven't mastered a skill. John recalls "guessing" his way through college and still getting good grades. You look at their models, and you think they're both being too extreme, so you propose a model where students guess but not as much. The three models are given in Table 2. Assume for this problem that there is a single true BKT model.

	Chris'	John's	Yours
L_0	2/3	0.25	0.5
T	0.1	0.1	0.1
G	0	0.5	0.3
S	0.1	0.1	0.1

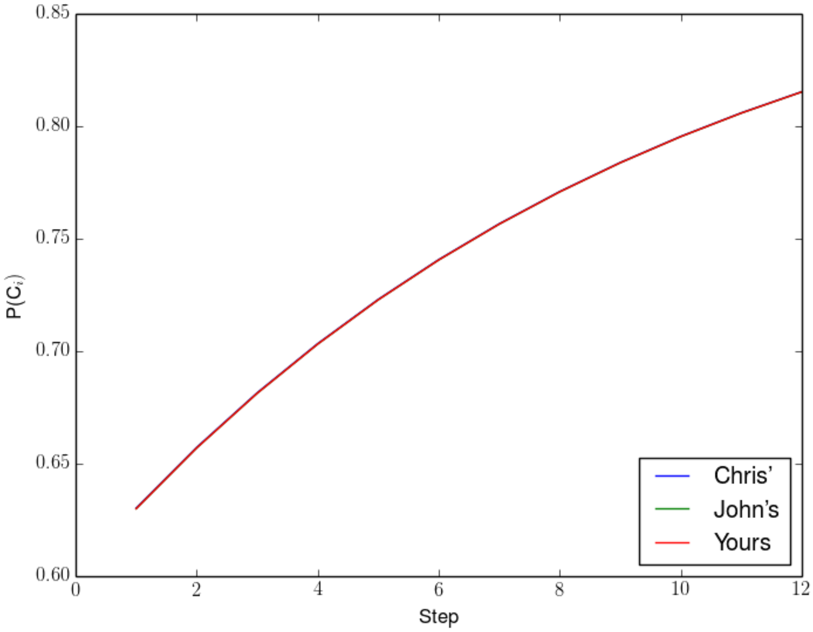
Table 2: Models provided by Chris, John, and You

No observations. First assume the student is given 7 problems but you have no observations of how the student does.

(a) Compute $P(K_i = 1)$ for $i = 0, \ldots, 12$ for each model and plot $P(K_i = 1)$ vs. i putting all three models on the same plot.



(b) Compute $P(C_i = 1)$ for $i = 1, \ldots, 12$ for each model and plot $P(C_i = 1)$ vs. i putting all three models on the same plot.

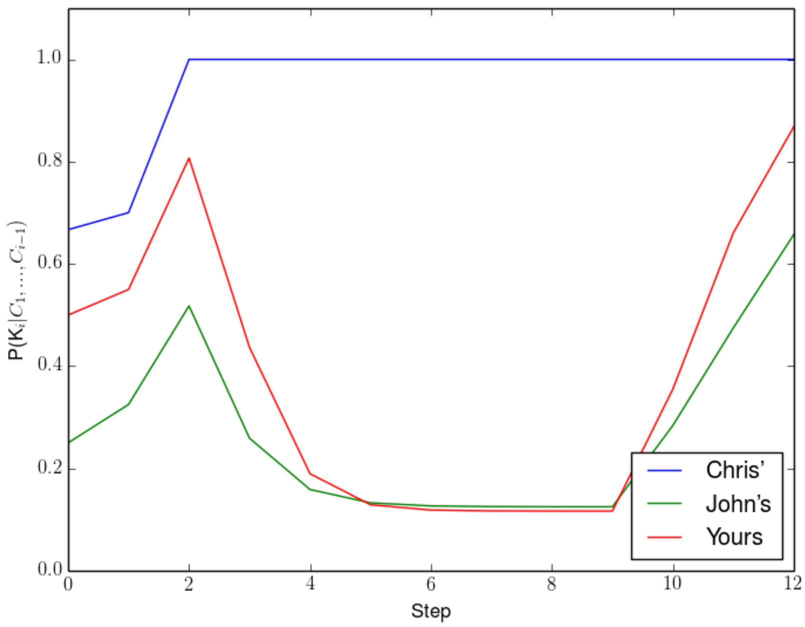


(c) Describe in 1-2 sentences the similarities and differences among the models in terms of their prediction of the student responses just computed.

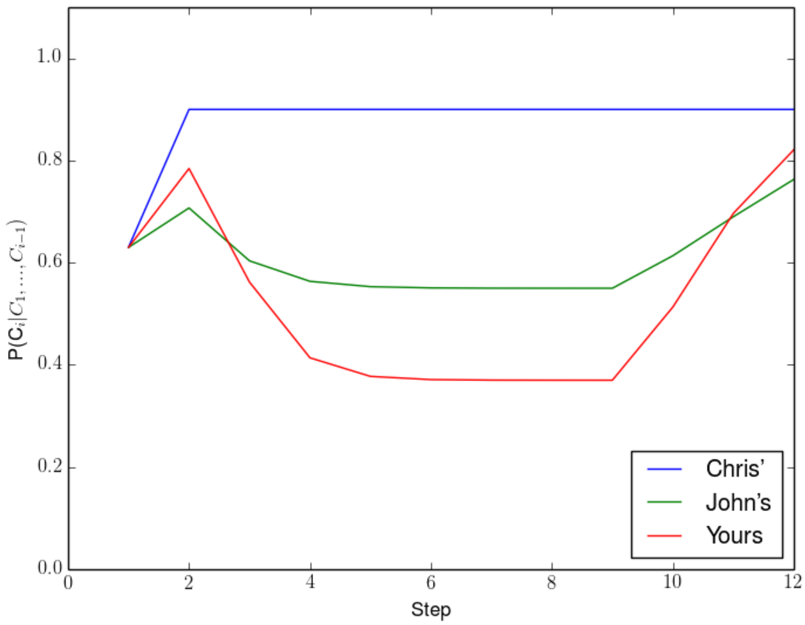
When we are predicting student responses ahead of time (i.e. before we have any observations), all the models appear identical! However, when we make predictions online (i.e. as we get observations), the models make drastically different predictions.

Adding observations Now suppose you collect some data for a single student and you see the following trajectory of observations $1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1$.

(d) Using the trajectory above, compute $P(K_i = 1|C_1, \ldots, C_{i-1})$ for $i = 0, \ldots, 12$ for each model and plot $P(K_i = 1|C_1, \ldots, C_{i-1})$ vs. i putting all three models on the same plot.



(e) Using the trajectory above, compute $P(C_i = 1|C_1, \ldots, C_{i-1})$ for $i = 1, \ldots, 12$ for each model and plot $P(C_i = 1|C_1, \ldots, C_{i-1})$ vs. i putting all three models on the same plot.



(f) Which model fits the data better in terms of log likelihood of the observations?

"Your model" fits the data the best.

(g) **Extra Credit** (2 pts): In (Beck and Chang, 2007), the authors use Table 1 and Figure

2 to claim that BKT models can be non-identifiable. Based on the results you saw in this question, do you agree with this claim? Why or why not?

No! The models are very identifiable. Even with one data point the models make very different predictions, and with more data points, we see that Chris' model is very different from the other two. "Your model" fits the data better than the other two. What the authors should have said is a priori the models make the same predictions; however, once you get data, one model will appear better than the others.

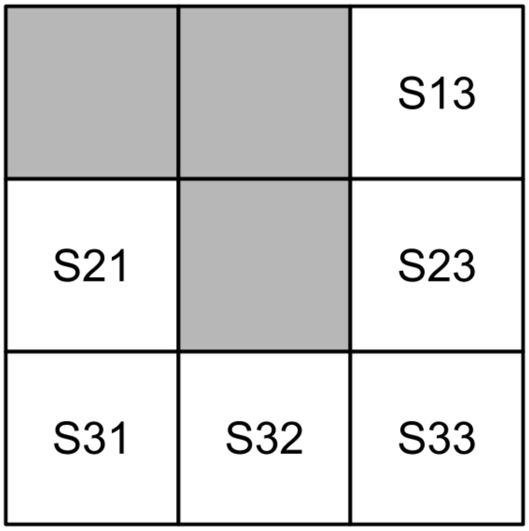
The authors of that paper are really concerned about another problem with BKT models—the model that fits the data the best, is not necessary the most semantically interpretable model. For example, the best fitting model might have a guess or slip parameter that is too large given the meanings we assign to the words "guess" and "slip". There are several ways to try to avoid this issue, and one is presented in that paper.

References

(Beck and Chang, 2007) Beck, J. E., & Chang, K. M. (2007). Identifiability: A fundamental problem of student modeling. In User Modeling 2007 (pp. 137-146). Springer Berlin Heidelberg.
(Corbett and Anderson, 1994) Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. User modeling and user-adapted interaction, 4(4), 253-278.

5 Navigation with POMDP

In this problem, we will explore using POMDPs to plan for navigating in discrete grid worlds, specifically as illustrated below.



The map of the world is known, but the robot's position and orientation is unknown. The goal for the robot is to get to square S31 and announce that it has reached the goal. If it announces and is actually at the goal location, it receives a reward of +100; if it announces and is not at the goal, the reward is -1000. After announcing, the robot is magically transported to a sink state (not illustrated) where all actions leave it in that state and it gets no further reward. The robot can also move forward, turn left, and turn right. Turning left or right is deterministic. Moving forward takes it to the square in front of it, unless that square is blocked by a wall, in which case the robot stays where it is. Walls are the outside boundary of the grid and the boundaries of the gray squares, and are represented as solid black lines. The robot can also observe what is in front of it at a cost of 10. The observation tells whether there is a wall in front with 90% accuracy.

(a) Write this domain as a POMDP. Include descriptions of the states, actions, transition function, observations, observation function, and rewards. Note that all actions and observation functions must be defined for all states. Use the 3×3 grid above as the environment that the robot needs to explore. For cases where the transition or observation functions are repetitive, you can just enumerate the functions for one set of states and describe the pattern for the other states.

States: There are (6*4)+1=25 states. We have a state for each combination pair of the 6 white grid cells and the 4 orientations and one for the sink state. sink, S13-N, S13-E, S13-S, S13-W, S21-N, S21-E, S21-S, S21-W, S23-N, S23-E, S23-S, S23-W, S31-N, S31-E, S31-S, S31-W, S32-N, S32-E, S32-S, S32-W, S33-N, S33-E, S33-S, S33-W

Actions: Forward, Left, Right, Observe, Announce

Transition function: In the following tuples of the form $< s, a, s >$, s is an element of S. A * in a given s is a wild card and it carries the same corresponding value in s.

<S13-N, Forward, S13-N> = 1.0
<S13-E, Forward, S13-E> = 1.0

<S13-S, Forward, S23-S> = 1.0
<S13-W, Forward, S13-W> = 1.0
<S13-N, Left, S13-W> = 1.0
<S13-E, Left, S13-N> = 1.0
<S13-S, Left, S13-E> = 1.0
<S13-W, Left, S13-S> = 1.0
<S13-N, Right, S13-E> = 1.0
<S13-E, Right, S13-S> = 1.0
<S13-S, Right, S13-W> = 1.0
<S13-W, Right, S13-N> = 1.0
<S13-, Observe, S13-*> = 1.0
<S13-, Announce, sink> = 1.0

The transition probabilities for the other states follow a similar pattern. Note specifically that since S13 is blocked in the N, E and W directions, the Forward action does not change state whereas in the S facing state, the Forward action changes state to S23-S. This mapping will be different for each grid cell depending on which directions are blocked. The Left, Right, Observer and Announce actions have the same transition probabilities for all states of any grid cell.

Observations: Wall, Open

Observation Function:
p(Wall | S13-N) = 0.9
p(Open | S13-N) = 0.1
p(Wall | S13-E) = 0.9
p(Open | S13-E) = 0.1
p(Wall | S13-S) = 0.1
p(Open | S13-S) = 0.9
p(Wall | S13-W) = 0.9
p(Open | S13-W) = 0.1

If the above p(o|s) were written more generally as p(o| <s,a,s>), a would always be the Observe action and s would be the same as s. Note that the observation probabilities are reversed for S13-S since there is no wall in front of the robot in that state. This pattern can be extended to all states according to the orientations blocked by walls. For all other actions (i.e., a != Observe), the probability of observing a wall is 0.5, since the action provides no additional information about the walls.

Rewards:
<*-*, Observe> = -10
<S31-*, Announce> = +100
<!S31-*, Announce> = -1000
Reward for all other <s,a> = 0

(b) Assume that the robots initial belief is uniformly distributed amongst all the white cells in the environment, but it knows it is facing north (up, in the diagram). Assume that the robot executes the sequence ⟨*forward, right, observe, left, forward, observe*⟩. Assume the first observation returns wall and the second returns open. Show the belief states of the robot after each action.

B0: S13-N: 1/6, S21-N: 1/6, S23-N: 1/6, S31-N: 1/6, S32-N: 1/6, S33-N: 1/6
B1: S13-N: 1/3, S21-N: 1/3, S23-N: 1/6, S32-N: 1/6
B2: S13-E: 1/3, S21-E: 1/3, S23-E: 1/6, S32-E: 1/6
B3: S13-E: 0.9 * 1/3, S21-E: 0.9 * 1/3, S23-E: 0.9 * 1/6, S32-E: 0.1 * 1/6
→ S13-E: 3/10, S21-E: 3/10, S23-E: 3/20, S32-E: 1/60
→ S13-E: 18/60, S21-E: 18/60, S23-E: 9/60, S32-E: 1/60
→ S13-E: 18/46, S21-E: 18/46, S23-E: 9/46, S32-E: 1/46 (Normalization)
→ S13-E: 9/23, S21-E: 9/23, S23-E: 9/46, S32-E: 1/46
B4: S13-N: 9/23, S21-N: 9/23, S23-N: 9/46, S32-N: 1/46
B5: S13-N: 27/46, S21-N: 9/23, S32-N: 1/46
B6: S13-N: 0.1 * 27/46, S21-N: 0.1 * 9/23, S32-N: 0.1 * 1/46
→ S13-N: 27/460, S21-N: 18/460, S32-N: 1/460
→ S13-N: 27/46, S21-N: 18/46, S32-N: 1/46 (Normalization)
→ S13-N: 27/46, S21-N: 9/23, S32-N: 1/46