CS 188 Spring 2006 Introduction to Artificial Intelligence Practice Final Sol'ns

1. (20 points.) True/False

Each problem is worth 2 points. Incorrect answers are worth 0 points. Skipped questions are worth 1 point.

(a) *True/False*: All MDPs can be solved using expectimax search.

False. MDPs with self loops lead to infinite expectimax trees. Unlike search problems, this issue cannot be addressed with a graph-search variant.

(b) *True/False*: There is some single Bayes' net structure over three variables which can represent any probability distribution over those variables.

True. A fully connected Bayes' net can represent any joint distribution.

(c) *True/False*: Any rational agent's preferences over outcomes can be summarized by a single real valued utility function over those outcomes.

True. Any set of preferences which conform to the six constraints on rational preferences (orderability, transitivity, continuity, substitutability, monotonicity, decomposability) can be summarized by a single, real-valued function.

(d) *True/False*: Temporal difference learning of optimal utility values (U) requires knowledge of the transition probability tables (T).

Mostly True. Temporal difference learning is a model-less learning technique that requires only example state sequences to learn the utilities for a fixed policy. However, to derive the best policy from those utilities, which would be required to find the optimal utility values, we would need to compute

$$\pi(s) = \arg\max_a \sum_{s'} T(s,a,s')U(s')$$

which of course includes a transition probability. The solution reads "mostly" true because the optimal utility values could be found without the transition probabilities if the agent were also supplied with the optimal policy. In practice, we could also estimate the transition probabilities from the training data (using maximum-likelihood estimates, for example), so they need not necessarily be known in advance.
NOTE: This solution was updated since the review session.

(e) *True/False*: Pruning nodes from a decision tree may have no effect on the resulting classifier.

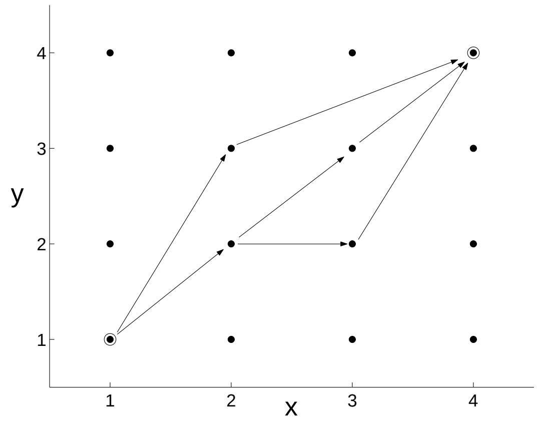
True. Trivially, a decision tree may have branches that are unreachable. Furthermore, splits in the decision tree may also refine $P(class)$, but have no effect in practice because of rounding. Imagine a leaf has 10 true, 3 false, and splits to 5/2 and 5/1 – you'll still guess true on each branch, but the split is refining the conditional probabilities.
NOTE: This solution was updated since the review session.

2. (20 points.) Search

Consider the following search problem formulation:

States: 16 integer coordinates, $(x, y) \in [1, 4] \times [1, 4]$
Initial state: $(1, 1)$
Successor function: The successor function generates 2 states with different y-coordinates
Goal test: $(4, 4)$ is the only goal state
Step cost: The cost of going from one state to another is the Euclidean distance between the points

We can specify a state space by drawing a graph with directed edges from each state to its successors:



Uninformed Search Consider the performance of DFS, BFS, and UCS on the state space above. Order successors so that DFS or BFS explores the state with lower y-coordinate first.

a) What uninformed search algorithm(s) find an *optimal* solution? What is this path cost?
Solution: UCS returns the lowest-cost path $((1, 1), (2, 2), (3, 3), (4, 4))$, of cost $3\sqrt{2}$.

b) What uninformed search algorithm(s) find a *shortest* solution? How long is this path?
Solution: BFS returns the path $((1, 1), (2, 3), (4, 4))$, of length 2.

c) What uninformed search algorithm(s) are most *efficient*? How many search nodes are expanded?
Solution: DFS expands 3 nodes corresponding to states $(1, 1), (2, 2), (3, 2)$. The goal node is not expanded.

Heuristic Search Use the Euclidean distance to the goal as a heuristic for A^* and greedy best-first search:

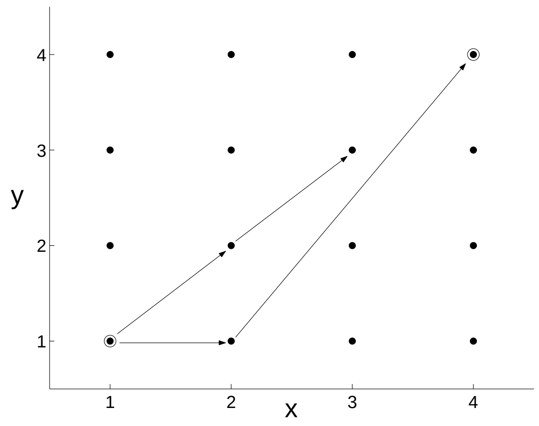
d) What heuristic search algorithm(s) find an *optimal* solution?
Solution: A^* search is optimal given an admissible heuristic such as Euclidean distance.

e) What heuristic search algorithm(s) find a *shortest* solution?
Solution: Greedy best-first search returns the path $((1, 1), (2, 3), (4, 4))$, of length 2.

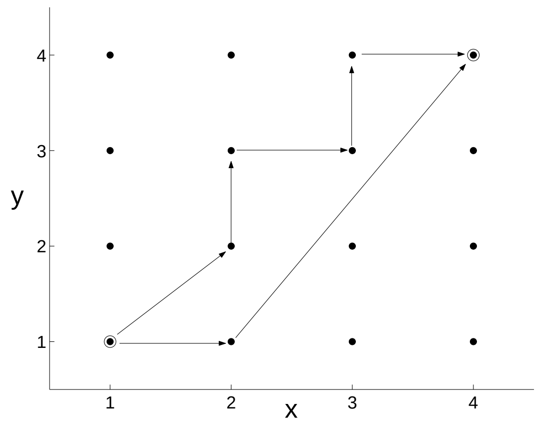
f) What heuristic search algorithm(s) are most *efficient*? How many search nodes are expanded?
Solution: Greedy best-first search expands 2 nodes corresponding to states $(1, 1), (2, 3)$.

In this family of grid-based problems, we can specify a state space by drawing a graph with directed edges from each state to its successors. Assume the search procedures order successors so that DFS or BFS explores the state with lower y-coordinate first.

a) Show a state space in which DFS and BFS are more efficient than A^* with the Euclidean heuristic.



b) Is the Manhattan distance to the goal an admissible heuristic for this search problem? If so, prove it; otherwise, show a state space in which A^* with the Manhattan heuristic is not optimal.



3. (20 points.) CSPs

[From a previous year's exam.] Consider the problem of tiling a surface (completely and exactly covering it) with n dominoes (each is a 2×1 rectangle, and the number of pips is irrelevant). The surface is an arbitrary edge-connected (i.e., adjacent along an edge, not just a corner) collection of $2n$ 1×1 squares (e.g., a checkerboard, a checkerboard with some squares missing, a 10×1 row of squares, etc.).

a) Formulate this problem precisely as a CSP where the dominoes are the variables (i.e., define the variable domain and the constraints).

Variables: $X_1, \ldots, X_n$
Domains:
The value of a domino variable is the pair of adjacent squares that it covers.
$\forall\ i \in \{1, \ldots, n\}$: $D_i = \{(s, s')\ :\ \text{square } s \text{ is adjacent to square } s'\}$
Constraints:
No two dominoes may cover the same square.
$\forall\ (i, j) \in \{1, \ldots, n\}^2$: with $i \neq j$ $C_{ij} = \{((s_i, s'_i), (s_j, s'_j))\ :\ |\{s_i, s'_i\} \cup \{s_j, s'_j\}| = 4\}$

NOTE: This solution was updated since the review session.

b) Formulate this problem precisely as a CSP where the squares are the variables (i.e., define the variable domain and the constraints). [*Hint: it doesn't matter which particular domino covers a given pair of squares.*]

Variables: $X_1, \ldots, X_{2n}$
Domains:
The value of a square is the adjacent square which is covered by the same domino.
$\forall\ i \in \{1, \ldots, 2n\}$: $D_i = \{s'_i\ :\ \text{square } s_i \text{ is adjacent to square } s'_i\}$
Constraints:
Adjacent squares must agree: square $X_i = s_j$ if and only if $X_j = s_i$.

c) For your formulation in part (b), describe exactly the set of instances that have a tree-structured constraint graph.

Solution: The constraints in (b) are binary constraints relating adjacent squares, so the constraint graph will form a loop whenever the squares form a loop (e.g., any 2×2 block). So any problem in which the "width" of the surface is 1 and no loops will have a tree-structured constraint graph (assuming it's one connected component). This property could easily be verified for a graph by performing a depth-first search from any node. If a node is reached exactly once in this manner (assuming a node's parent is not one of its successors), then the constraint graph will be a tree.

d) [Extra credit] Consider domino-tiling a checkerboard in which two opposite corners have been removed (31 dominoes to tile 62 squares). Prove that the CSP has no consistent assignment.

Solution: Consider the alternating colors of checkerboard squares: no two adjacent tiles have the same color, thus a domino will always cover two tiles of different colors. If we remove opposite corners of the board, there are 30 squares of one color and 32 squares of another color. Hence, it is not possible for 31 dominoes to each cover one square of each color.

4. (20 points.) Probabilistic Reasoning

Consider the problem of predicting a coin flip on the basis of several previous observations. Imagine that you observe three coin flips, and they all come up heads. You must now make a prediction about the (distribution of the) outcome of a fourth flip.

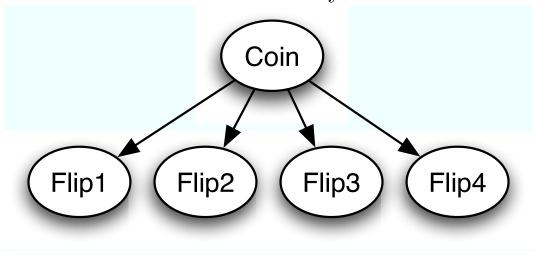
a) What is the maximum likelihood estimate for the probability that the fourth flip will be heads?

Solution:$P(heads) = 1$ gives a likelihood of 1 to the observation.

Assume that there are three kinds of coins in the world: fair coins, two-headed coins which always come up heads, and two-tailed coins which always come up tails. Moreover, assume that you have a prior belief or knowledge that a given coin is fair with probability 0.5, and two-headed or two-tailed with probability 0.25 each.

b) Draw a Bayes' net which expresses the assumptions that a single coin is chosen, then the flip outcomes are indentically distributed and conditionally independent given the coin type. Your network should include a node for the unobserved fourth flip.

Solution: The Bayes' net has the structure of a naive Bayes model:



c) What is the posterior probability that the coin is fair given the three heads observations?

Solution: The posterior $P_{c|f} = P(Coin = Fair|Flip1 = H, Flip2 = H, Flip3 = H)$ is calculated using Bayes' rule.

$$\begin{aligned}
P_{c|f} &= P(Coin = Fair|Flip1 = H, Flip2 = H, Flip3 = H) \\
&= \frac{P(Coin = Fair)P(Flip1 = H, Flip2 = H, Flip3 = H|Coin = Fair)}{P(Flip1 = H, Flip2 = H, Flip3 = H)} \\
&= \frac{P(Coin = Fair)P(Flip1 = H, Flip2 = H, Flip3 = H|Coin = Fair)}{\sum_c P(Coin = c)P(Flip1 = H, Flip2 = H, Flip3 = H|Coin = c)} \\
&= \frac{0.5 \cdot \frac{1}{8}}{0.5 \cdot \frac{1}{8} + 0.25 \cdot 1 + 0.25 \cdot 0} \\
&= \frac{1}{5}
\end{aligned}$$

d) What is the probability that the fourth flip will be heads given the three heads observations?

Solution: We begin by stating that the probability of $Flip4$ can be expressed as a sum over the joint probability of $(Coin, Flip4)$, which can be expanded using the chain rule.

$$P(Flip4) = \sum_{coin} P(coin)P(Flip4|coin)$$

This relationship is also true conditioned on the evidence:

$$P(Flip4|Flip1, Flip2, Flip3) = \sum_{coin} P(coin|Flip1, Flip2, Flip3)P(Flip4|Flip1, Flip2, Flip3, coin)$$

But by the conditional independence stated in (c), the final term is equal to $P(Flip4|coin)$. Hence,

$$\begin{aligned} P(Flip4 = H|Flip1, Flip2, Flip3) &= \sum_{c} P(Coin = c|Flip1, Flip2, Flip3)P(Flip4 = H|Flip1, Flip2, Flip3, Coin = c) \\ &= \sum_{c} P(Coin = c|Flip1, Flip2, Flip3)P(Flip4 = H|Coin = c) \\ &= \frac{1}{5} \cdot \frac{1}{2} + \frac{4}{5} \cdot 1 = \frac{9}{10} \end{aligned}$$

Where the posteriors $P(Coin = c|Flip1, Flip2, Flip3)$ come from part (c). Intuitively, we are just computing an expectation that $Flip4$ will be heads given the posterior distribution over $Coin$ that we learned from part (c).

5. (20 points.) Tree Search

[Adapted from a previous exam] In the following, a "max" tree consists only of max nodes, whereas an "expectimax" tree consists of a max node at the root with alternating layers of chance and max nodes.

(a) Assuming that leaf values are finite but unbounded, is pruning (as in alpha-beta) ever possible in a max tree? Give an example, or explain why not.

Solution: No pruning is possible. The next branch of the tree may always contain a higher value than any seen before.

(b) Is pruning ever possible in an expectimax tree under the same conditions? Give an example, or explain why not.

Solution: No pruning is possible. Again, the next branch of the tree may always contain a higher value than any seen before at a max node. The expectation node must be computed in its entirety as well for the max node above it.

(c) If leaf values are constrained to be nonnegative, is pruning ever possible in a max tree? Give an example, or explain why not.

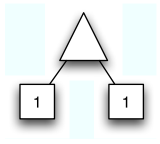
Solution: No pruning is possible. Establishing a lower bound does not address the issue in (a).

(d) If leaf values are constrained to be nonnegative, is pruning ever possible in an expectimax tree? Give an example, or explain why not.

Solution: No pruning is possible. Establishing a lower bound does not address the issue in (b).

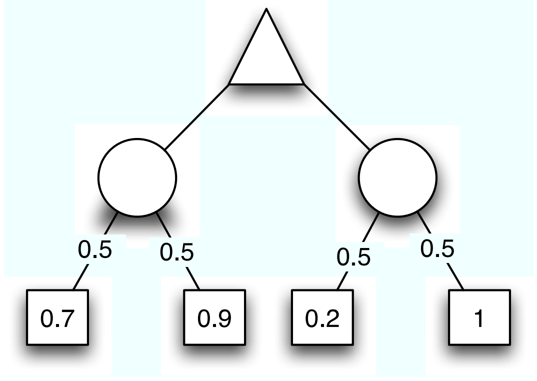
(e) If leaf values are constrained to be in the range $[0, 1]$, is pruning ever possible in a max tree? Give an example, or explain why not.

Solution: The second leaf can be pruned in:



(f) If leaf values are constrained to be in the range $[0, 1]$, is pruning ever possible in an expectimax tree? Give an example (qualitatively different from your example in (e), if any), or explain why not.

Solution: Assuming a left-to-right ordering, the right-most leaf can be pruned in:



(g) Consider the outcomes of a chance node in an expectimax tree. Which of the following orders is most likely to yield pruning opportunities?

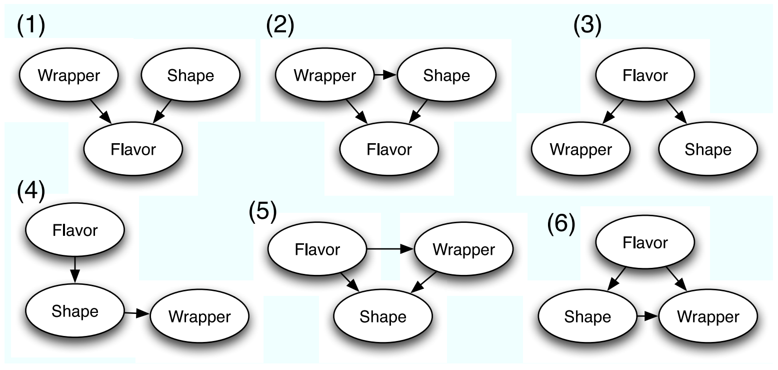
- Lowest probability first
- Highest probability first
- Doesn't make a difference

Solution: Highest probability first.

6. (20 points.) Bayes' Nets

[Adapted from a previous exam] The Surprise Candy Company makes candy in two flavors: 30% are Anchovy (which smell of the salty sea) and 70% are Berry (which smell delightful. All candies start out round and look the same. Someone (who can smell) trims some of the candies so that they are square. Then, a second person who *can't* smell wraps each candy in a red or brown wrapper. 80% of Berry candies are round and 74% have red wrappers. 90% of Anchovy candies are square and 82% have brown wrappers.

All candies are sold individually in sealed, identical, black boxes! You have just bought a box.



(a) Which network(s) can correctly represent $P(Flavor, Wrapper, Shape)$?

Solution: 2, 4, 5, 6. Bayes net 3 is not correct because it asserts that the shape and wrapper are independent given the flavor. However, in the project description, we see that the wrapper is chosen using only information about the shape (not the flavor), so they are not independent.

NOTE: This solution was updated since the review session.

(b) Which network(s) assert(s) $P(Wrapper|Shape) = P(Wrapper)$?

Solution: 1

(c) Which network(s) assert(s) $P(Wrapper|Shape, Flavor) = P(Wrapper|Shape)$?

Solution: 4

(d) From the problem description, what independence relationships should hold for this problem? Which network is the best representation of this problem?

Solution: 4 asserts the correct independence relation: $P(Wrapper|Shape, Flavor) = P(Wrapper|Shape)$

(e) What is the probability that the candy has a red wrapper?

$$P(Wrapper = Red) = \sum_{flavor} P(flavor)P(Wrapper = Red|flavor) = .7 \cdot .74 + .3 \cdot .18 = .572$$

NOTE: This solution was updated since the review session.

(f) In the box is a round candy with a red wrapper. What is the probability that it is a Berry candy?

Solution: The flavor is independent of the wrapper given the shape. So, we need only use Bayes' rule as follows:

$$P(berry|round) = \frac{P(round|berry)P(berry)}{\sum_{flavor} P(round|flavor)P(flavor)}$$

Filling in the numbers, we have:

$$P(berry|round) = \frac{0.8 \cdot 0.7}{0.8 \cdot 0.7 + 0.1 \cdot 0.3} = \frac{56}{59}$$

(g) If you tried to model the problem with network (2), can you give $P(Wrapper = Red|Shape = Round)$? If so, give the answer.

Solution: The network in (2) can represent any probability distribution over the three variables. However, probabilities given in the question underspecify this joint probability distribution, so we don't have enough information to fill in the CPTs for (2).

(h) If you tried to model the problem with network (4), can you give $P(Wrapper = Red|Shape = Round)$? If so, give the answer.

Solution: The network in (4) specifies only one distribution that is consistent with the probabilities in the problem description. Finding this solution is quite tricky.

First, we observe that the network in (4) is a Markov chain. Recalling the forward algorithm we can see that**

$$P(Wrapper|Flavor) = \sum_{shape} P(shape|Flavor)P(Wrapper|shape)$$

This computation amounts to summing the probabilities of all the paths that get from $Flavor$ to $Wrapper$. Let's now specialize this equation for red wrappers:

$$\begin{aligned} P(red|berry) &= P(round|berry)P(red|round) + P(square|berry)P(red|square) \\ P(red|anchovy) &= P(round|anchovy)P(red|round) + P(square|anchovy)P(red|square) \end{aligned}$$

We can fill in all but two of these quantities given the data provided, which leaves us with a system of two linear equations with two unknowns.

$$\begin{aligned} 0.74 &= 0.8 \cdot P(red|round) + 0.2 \cdot P(red|square) \\ 0.18 &= 0.1 \cdot P(red|round) + 0.9 \cdot P(red|square) \end{aligned}$$

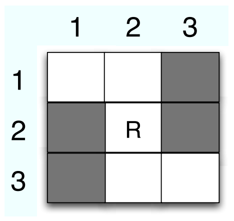
Solving these equations simultaneously, we have $P(red|round) = 0.9, P(red|square) = 0.1$.

** To see why this Markov chain property is true from first principles, observe that

$$\begin{aligned} P(Wrapper|Flavor) &= \frac{P(Wrapper, Flavor)}{P(Flavor)} \\ &= \frac{\sum_{shape} P(Wrapper, Shape, Flavor)}{P(Flavor)} \\ &= \frac{\sum_{shape} P(Flavor)P(Shape|Flavor)P(Wrapper|Shape)}{P(Flavor)} \\ &= \sum_{shape} P(Shape|Flavor)P(Wrapper|Shape) \end{aligned}$$

This question proved trickier than we expected. Nothing this involved is going to appear on the actual exam. *NOTE:* This solution was updated since the review session.

7. (20 points.) HMMs



Robot Localization Grid

Suppose you are a robot navigating a maze (see figure 1), where some of the cells are free and some are blocked. At each time step, you are occupying one of the free cells. You are equipped with sensors which give you noisy observations, (w_U, w_D, w_L, w_R) of the four cells adjacent to your current position (*up,down,left*, and *right* respectively). Each w_i is either *free* or *blocked*, and is accurate 80% of the time, independently of the other sensors or your current position. [1]

Imagine that you have experienced a motor malfunction that causes you to randomly move to one of the four adjacent cell with probability $\frac{1}{4}$.[2]

a) Suppose you start in the central cell in figure 1. One time step passes and you are now in a new, possibly different state and your sensors indicate (*free,blocked,blocked,blocked*). Which states have a non-zero probability of being your new position?

Solution:$(2,1), (2,2), (2,3)$

b) Give the posterior probability distribution over your new position.

Solution: The posterior probabilities of each state are proportional to the probability of ending up in the state after one random move weighted by the probability of the observation from that state:

$$\begin{aligned}
P(s_1 = (2,1)|w_U, w_D, w_L, w_R) &\propto \frac{1}{4} \cdot 0.8^1 \cdot 0.2^3 = 0.0016 \\
P(s_1 = (2,2)|w_U, w_D, w_L, w_R) &\propto \frac{1}{2} \cdot 0.8^3 \cdot 0.2^1 = 0.0512 \\
P(s_1 = (2,3)|w_U, w_D, w_L, w_R) &\propto \frac{1}{4} \cdot 0.8^3 \cdot 0.2^1 = 0.0256
\end{aligned}$$

Normalizing, we have

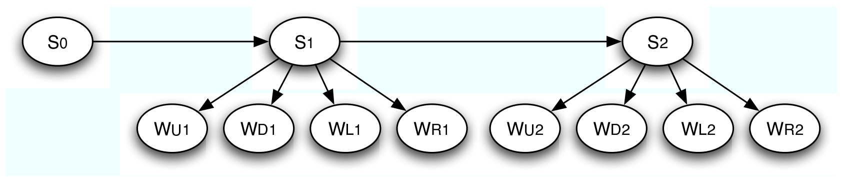
$$\begin{aligned}
P(s_1 = (2,1)|w_U, w_D, w_L, w_R) &= 0.02 \\
P(s_1 = (2,2)|w_U, w_D, w_L, w_R) &= 0.65 \\
P(s_1 = (2,3)|w_U, w_D, w_L, w_R) &= 0.33
\end{aligned}$$

This distribution matches our intuition: a move upward was quite unlikely because of the observation, which disagrees with the layout around $(2,1)$ in three out of four directions.

c) Suppose that s_0 is your starting state and that s_1 and s_2 are random variables indicating your state after the first and second time steps. Draw a Bayes' net illustrating the relationships between each s_i and the sensor observations associated with that state. Consider the CPT for s_1. How many values for s_1 have non-zero probability?

[1]Assume that if a cell is off the given grid it is treated as blocked.
[2]If you move towards a blocked cell, you hit the wall and stay where you are.

Solution:



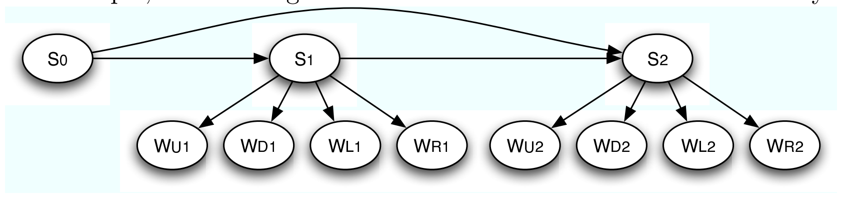
d) Suppose that $p_t(s)$ is a function representing the proability that at time step t, you are in state s. Using your bayes net diagram from (c), write a recursive function for $p_{t+1}(s)$ in terms of $p_t(s)$ and the observations (w_U, w_D, w_L, w_R) for time step $t+1$. Make sure your formulation is general and not specific to the particular grid in figure 1.

Solution: To write a function for the probability of ending up in state s' given state s that you were in before, we take a sum of the product of the transition and observation probabilities the normalize, just like the forward algorithm for HMMs.[3]

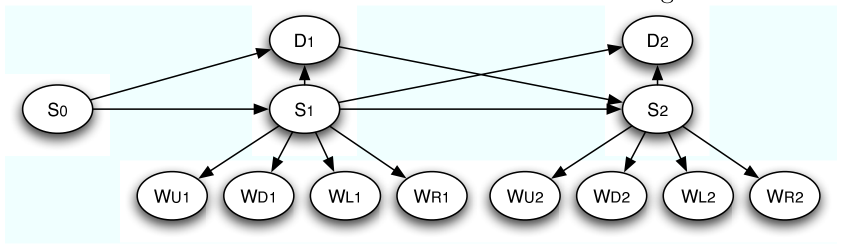
$$p_{t+1}(s') = \frac{\sum_s p_t(s)P(S_{t+1} = s'|S_t = s)\prod_{w\in\{w_U,w_D,w_L,w_R\}} P(w|s')}{\sum_{s''}\sum_s p_t(s)P(S_{t+1} = s''|S_t = s)\prod_{w\in\{w_U,w_D,w_L,w_R\}} P(w|s'')}$$

e) Now suppose that your motor is not malfunctioning, but is instead being controlled by a person. You have no information about what your controller intends for your path to be, but you do know that your movements tend to continue in a given direction for a while before changing directions. More specifically, assume that at each time step, you will either be moved in the same direction as the previous times step (perhaps bouncing you into a wall) or that you will move in a random new direction (possibly resulting in the same direction again). Draw a bayes net representing this situation for three time steps i.e s_0, s_1, s_2. Make sure to briefly describe any new variables you introduce into the diagram. Indicate (in at most one sentence) how this preference for going in the same direction will be incoporated into the CPT of the nodes in your net.

Solution: Two solutions make sense to model this behavior. First, we can make s_2 depend on both s_1 and s_0. In this case, the CPT for s_2 will incorporate the bias to continue in the same direction as before. While this structure might seem simple, determining the values of the CPT tables would be fairly involved.



Alternatively, we can introduce a new variable that encapsulates the direction of movement, denoted D_i in the figure below. This variable would take 5 values: one for each direction or a special value nd to indicate that the next state will not be determined by the previous direction. Then, the CPT for s_2 would be deterministic given one of the first four values and the same values as it had in the original network when $d_1 = nd$.



[3]This question is perhaps a bit ambiguous. Some may interpret it as asking for a function for the probability of staying in the same state you were in before, which is a special case of the solution given.

8. (20 points.) MDPs and Reinforcement Learning

[Adapted from Sutton and Barto] A cleaning robot must vacuum a house on battery power. It can at any time either *clean* the house, *wait* and do nothing, or *recharge* its battery. Unfortunately, it can only perceive its battery level (its state) as either *high* or *low*. If the robot recharges, the battery return to *high*, and the robot receives a reward of 0. If the robot waits, the battery level is unchanged, and the robot receives a reward of R_{wait}. If the robot cleans, the outcome will depend on the battery level. If the battery level is high, the battery drops to low with fixed probability 1/3. If the battery level is low, it runs out with probability 1/2. If the battery does not run out, the robot receives a reward of R_{clean}. If the battery does run out, a human must collect the robot and recharge it. In this case, the robot ends up with a high battery, but receives a reward of -10.

Note that the robot receives rewards not based on its current state (as in the book), but based on state-action-state triples (as in project 4). For this problem, you may assume a discount rate of 0.9.

a) Assuming $0 \leq R_{wait} \leq R_{clean}$, which of the following can be optimal behavior (circle all that apply).

(a) Always *clean*, regardless of battery.
Solution: If R_{clean} is large enough, this will be the optimal policy.

(b) Always *recharge*, regarless of battery.
Solution: If $R_{clean} = R_{wait} = 0$, this will be the optimal policy.

(c) *Clean* with high battery, *recharge* with low battery.
Solution: For some reasonably high rewards, this is the policy we'd expect.

(d) *Recharge* with high battery, *clean* with low battery.
Solution: This *cannot* be the optimal policy because if cleaning with low battery is optimal, then cleaning with high battery will be optimal.

b) Write a Bellman equation relating the optimal utility of the state *low* in terms of other optimal utilities. You should use specific variables and probabilities from the problem statement above when possible.

*Solution:*With the reward structure for state-action-state triples, we most naturally consider the optimal utility to be the discounted sum of all *future* rewards.

$$U^*(low) = \max\left(\gamma U^*(high), R_{wait} + \gamma U^*(low), \frac{1}{2}(R_{clean} + \gamma U^*(low)) + \frac{1}{2}(-10 + \gamma U^*(high))\right)$$

where $\gamma = 0.9$, as mentioned in the problem description.

c) Let $R_{clean} = 3$ and $R_{wait} = 1$. Assume you have an initial estimate of zero for each state's utility. What are the estimates after one round of value iteration?

*Solution:*After 1 round of value iteration, the states take their maximal expected rewards after one action since future rewards are 0. Thus, we have

$$U(high) = 3, U(low) = 1$$

d) Using the rewards above and a learning rate of 0.2, consider using Q-learning to estimate q-values for this problem. Assume all of your q-value estimates are zero initially. Imagine you observe the following sequence of states, actions, and rewards: high, clean, +3, high, clean, +3, low, clean +3, low, clean -10. Show the q-values for each state after each reward.

Solution: The Q update for this problem is precisely what we saw for the robot project. We need only perform those updates to solve the problem:

After high, clean, +3, we have $Q(high, clean) = .6$ and all other Q are 0.

After another high, clean, +3, we have $Q(high, clean) = .8 \cdot .6 + .2(3 + 0.9 \cdot 0) = 1.08$ because the next state is *low*, and all other Q are 0.

After low, clean +3, we have $Q(high, clean) = 1.08, Q(low, clean) = 0.6$ and all other Q are zero.

After low, clean -10, we have $Q(low, clean) = 0.8 \cdot 0.6 + 0.2(-10 + 0.9 \cdot 1.08) = -1.33$. We can make the Q update because we know that the following state will be *high*. Note that the agent would not be able to make the update until it identified its next state to be *high* and therefore could compute $\max_{a'} Q(high, a') = 1.08$. After this final update, we would still have Q(high, clean) = 1.08 and all other Q equal to 0.