CS221 Practice Solutions #1

Summer 2013

The following pages are excerpts from similar classes' midterms. The content is similar to our midterm – but I have opted to give you a document with more problems rather than one that follows the structure of the midterm precisely. See the midterm handout for more details on what the exam will look like. The midterm is 2 hours. It is open book and open computer but **closed** Internet.

1. [Short Answers] Truth Test (10 points)

For the following questions, a correct answer is worth 2 points, no answer is worth 1 point, and an incorrect answer is worth 0 points. Circle true or false to indicate your answer.

a. (true or false) If g(s) and h(s) are two admissible A* heuristics, then their average f(s) = ½ g(s) + ½ h(s) must also be admissible.

True. Let h*(s) be the true distance from s. We know that g(s) <= h*(s) and h(s) <= h*(s), thus f(s) <= ½ h*(s) + ½ h*(s). We can simplify to f(s) <= h*(s).

b. (true or false) For a search problem, the path returned by uniform cost search may change if we add a positive constant C to every step cost.

True. Consider that there are two paths from the start state (S) to the goal (G), S → A → G and S → G. So the optimal path is through A. Now, if we add 2 to each of the costs, the optimal path is directly from S to G. Since uniform cost search finds the optimal path, its path will change.

c. (true or false) The running-time of an efficient solver for tree-structured constraint satisfaction problems is linear in the number of variables.

True. The running time of the algorithm for tree-structured CSPs is $O(n \cdot d^2)$, where n is the number of variables and d is the maximum size of any variable's domain.

d. (true or false) The amount of memory required to run minimax with alpha-beta pruning is O(bd) for branching factor b and depth limit d.

True and False (everyone wins). The memory required is only O(bd), so we accepted False. However, by definition an algorithm that is $O(bd)$ is also $O(b^d)$, because O denotes upper bounds that may or may not be tight, so technically this statement is True (but not very useful).

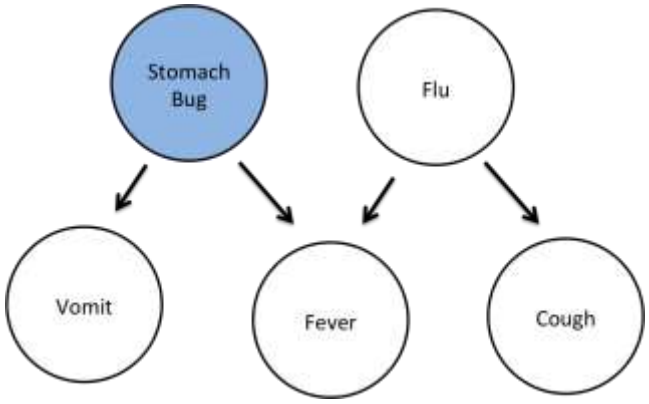
e. (true or false) For a discrete bayesian network with ***n*** variables, the amount of space required to store the "joint" distribution table is O(n).

False. The size of the joint is $O(d^n)$ where d is the domain of the variables.

f. (true or false) In a markov decision problem, there are no actions and instead everything is controlled by “chance.”

False. Agents make actions, the successor states of those actions are controlled by chance.

h. (true or false) Assume the Bayes Net from class is a prefect representation of the world:



You know that you *do not* have a stomach bug. If you were to vomit, would that information change the probability that you had a fever?

False. The two variables are conditionally independent given stomach bug (notably, the wording of this question didn’t sound true or false).

.

2. [Deterministic Search] Mr. and Ms. Pacman (20 points)

Pacman and Ms. Pacman are lost in an NxN maze and would like to meet; they don't care where. In each time step, both simultaneously move in one of the following directions: {NORTH, SOUTH, EAST, WEST, STOP}. They do not alternate turns. You must devise a plan which positions them together, somewhere, in as few time steps as possible. Passing each other does not count as meeting; they must occupy the same square at the same time.

a. Formally state this problem as a single-agent state-space search problem.

   States:

   The set of pairs of positions for Pacman and Ms. Pacman: {((x1, y1), (x2, y2)) | x1, x2, y1, y2 are in {1, 2, …, N}}

   Goal test:

   isGoal((x1, y1), (x2, y2)): return x1 == x2 and y1 == y2

   Legal actions (given a state):

   legalActions((x1, y1), (x2, y2)): If not blocked from their current positions, both pacman and mrs pacman can move north, south, east, west. They can always stop.

   Successor function (given a state and an action):

   successor(((x1, y1), (x2, y2)), action): Move pacman and mrs pacman from their current state in the direction they both moved (respectively). Return the new x1, y1, x2, y2 positions.

b. Give a non-trivial admissible heuristic for this problem.

Answer: Manhattan distance between Pacman and Ms. Pacman DIVIDED BY 2 (since both take a step simultaneously)

c. Circle all of the following graph search methods which are guaranteed to output optimal solutions to this problem:

(i) DFS (ii) BFS (iii) UCS

(iv) A* (with a consistent and admissible heuristic)

(v) A* (with heuristic that returns zero for each state)

Everything but DFS.

d. If h1 and h2 are admissible, which of the following are also guaranteed to be admissible? Circle all that apply:

(i) $h1 + h2$

(ii) $h1 * h2$

(iii) $\max(h1 , h2)$

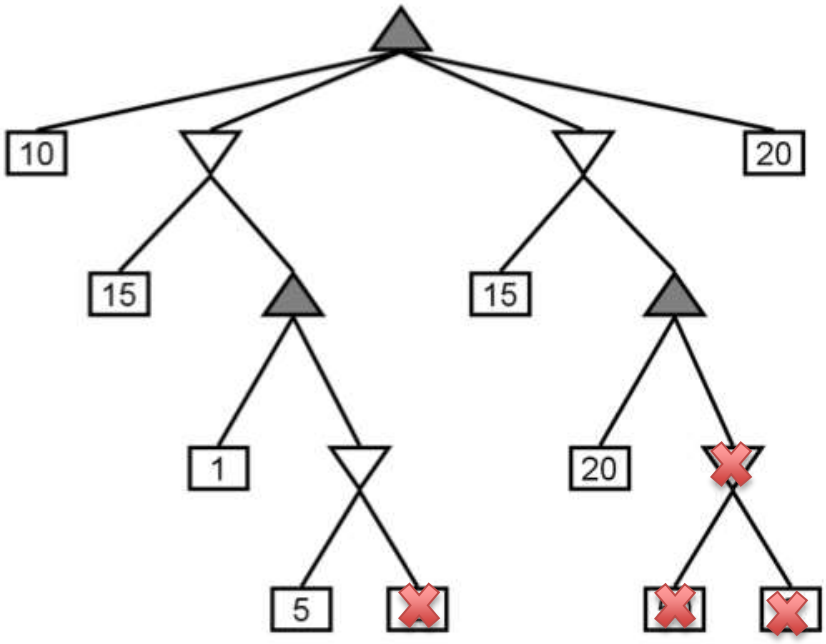
(iv) $\min(h1 , h2)$

(v) $(\alpha)h1+(1-\alpha)h2$ for any value α between 0 and 1

Answer: (iii), (iv), (v)

3. [Adversarial Search] MiniMax (15 points)

Consider the following minimax tree:



a. What is the minimax value for the root?

20

b. Draw an X through any nodes which will not be visited by alpha-beta pruning, assuming children are visited in left-to-right order.

See pic above.

c. Is there another ordering for the children of the root for which more pruning would result? If so, state the order.

Yes, if we had the children ordered as 20, 15, 10, 2.

[continued on the next page]

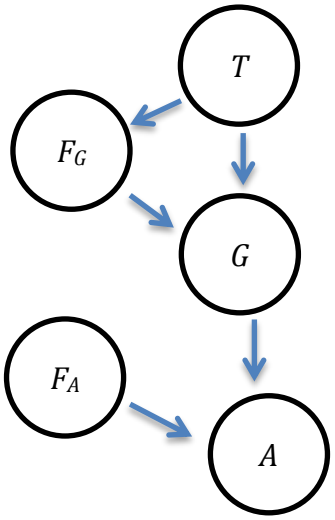
d. Propose a general, practical method for ordering children of nodes which will tend to increase the opportunities for pruning. You should be concise, but clearly state both what to do about min nodes and max nodes.

In general we would want to use an evaluation function to estimate the values of the children and then order them so that at max nodes we order the children with larger estimated values first and at min nodes we order the children with larger estimated values first. For those who did not include the evaluation function: ordering nodes by their true values is not practical.

4. [Bayes Net] Nuclear Power Station (15 points)

In your local nuclear power station, there is an alarm that senses when a temperature gauge exceeds a given threshold. The gauge measures the temperature of the core. Consider the Boolean variables ***A*** (alarm sounds), F_A (alarm is faulty), and F_G (gauge is faulty) and the multivalued, discrete nodes ***G*** (gauge reading) and ***T*** (actual core temperature).

a. Draw a Bayesian network for this domain, given that the gauge is more likely to fail when the core temperature gets too high.



b. Suppose there are just two possible actual and measured temperatures, normal and high; the probability that the gauge gives the correct temperature is ***x*** when it is working, but ***y*** when it is faulty. Give the conditional probability table associated with ***G***

	F_G = True		F_G = False	
	T= High	T = Normal	T = High	T= Normal
G = High	y	(1 – y)	x	(1 – x)
G = Normal	(1 – y)	y	(1 – x)	x

c. Suppose the alarm works correctly unless it is faulty, in which case it never sounds. Give the conditional probability table associated with ***A***

Alarm works correctly unless it is faulty. So, if alarm is not faulty, then the alarm works correctly, meaning that it sounds only if gauge reading is high.

	G = Normal		***G = High***	
	***F_A* = True**	***F_A* = False**	***F_A* = True**	***F_A* = False**
A = True	***0***	***0***	***0***	***1***
A = False	***1***	***1***	***1***	***0***

d. Suppose the alarm and gauge are working and the alarm sounds. Calculate an expression for the probability that the temperature of the core is too high, in terms of the various conditional probabilities in the network.

The assumption says that the alarm and gauge are working, which means that FA = :fa and FG = :fg. Also, we have that the alarm sounds (A = a). As we are asked the probability of the temperature being too high (T = t), we can formulate the query as:

$$P(t|a, \neg f_a, \neg f_g)$$

Which we would have to evaluate for the values of G (the remaining variable in our network). However, since we are using the assumptions above (also from parts (c) and (d)) we can deduce the following: Since the alarm sounds (a) and the alarm is not faulty, we can infer by watching Table that for this to be possible, the gauge reading must be high (G = g) Hence, we can rewrite the above expression as:

$$P(t|a, \neg f_a, g, \neg f_g)$$

First, we use Bayes'rule:

$$P(t|a, \neg f_a, g, \neg f_g) = \frac{P(t,\ a, \neg f_a, g, \neg f_g)}{P(\ a, \neg f_a, g, \neg f_g)}$$

Applying chain's rule up and down

$$P(t|a, \neg f_a, g, \neg f_g) = \frac{P(a|\neg f_a, g)P(\neg f_a)P(g|\neg f_g, t)P(\neg f_g|\ t)P(t)}{P(a|\neg f_a, g)P(\neg f_a)P(g, \neg f_g)\ P(\neg f_g)}$$

Eliminating common terms

$$P(t|a, \neg f_a, g, \neg f_g) = \frac{P(g|\neg f_g, t)P(\neg f_g|\ t)P(t)}{P(g, \neg f_g)P(\neg f_g)}$$

(PS. The calculations above could have been saved, because we all know that a and fa do not have direct relation with t. We just wanted to verify that this held true...and it did!)

Back to the problem, we can use the product version of the Bayes'rule for the denominator:

$$P(t|a, \neg f_a, g, \neg f_g) = \frac{P(g|\neg f_g, t)P(\neg f_g|\ t)P(t)}{P(g, \neg f_g)}$$

Since we have in Table the relation of g based on T and FG, we will try to use it. Again, in the denominator we use the variable T:

$$P(t|a, \neg f_a, g, \neg f_g) = \frac{P(g|\neg f_g, t)P(\neg f_g|\ t)P(t)}{P(g, \neg f_g, t) + P(g, \neg f_g, \neg t)}$$

And apply chain rule (guess where? Yes, the denominator still)

$$P(t|a, \neg f_a, g, \neg f_g) = \frac{P(g|\neg f_g, t)P(\neg f_g|\ t)P(t)}{P(g|\neg f_g, t)P(\neg f_g|t)P(t) + P(g|\neg f_g, \neg t)P(\neg f_g|\neg t)P(\neg t)}$$

Replacing what we have got:

$$P(t|a, \neg f_a, g, \neg f_g) = \frac{x\ P(\neg f_g|\ t)P(t)}{x\ P(\neg f_g|t)P(t) + (1-x)P(\neg f_g|\neg t)P(\neg t)}$$

To make things less messy, we call $P(t) = m$, $P(f_g \mid t) = n$ and $P(f_g \mid t) = q$,

so we have:

$$P\left(t\middle|a,\neg f_a,g,\neg f_g\right)=\frac{x\ (1-n)m}{x\ (1-n)m+(1-x)(1-q)(1-m)}$$

Which is our answer to this last item.

5. [Temporal Models] Knowledge Tracing (25 points)

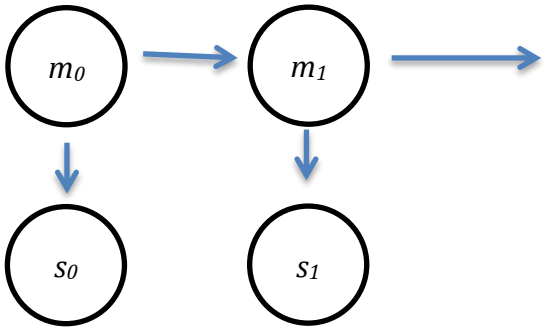
A new online education website wants to provide individualized education. For simplicity imagine that users log into the website and continually answer randomly generated questions that cover triangle inequality. The only information accessible by the website is the score that the student gets on the problems they solve. Your task is to use these scores to track the extent to which a given student has mastered triangle inequality.

(a) Is this problem better suited for a markov model or a hidden markov model?

Hidden Markov Model: The extent to which the student has mastered triangle inequality is a hidden variable.

(b) Formalize this real world problem:

(i) Draw the temporal Bayesian network.



(ii) For each variable in your model give a brief description of its role and specify a ***discrete*** domain.

m_i represents the students mastery at time i. Its domain could be any reasonable discrete domain. We chose {low, medium, high}

s_i represents the students score on problem i. Its domain could be any reasonable discrete domain. We chose {low, medium, high}

(iii) For each ~~arc~~ node in your model what are the dimensions of the CPT.

m_0 has CPT size 3
m_i has CPT size 9
s_i has CPT size 9

(c) We have invented a brain-measuring device that can measure exactly how well a student knows triangle inequality (You can call

```
getMastery()
```

at any point and the function will return a string in the domain {'low', 'medium', 'high'}).

Your task is to write python code that could use this device to learn the emission conditional probability table of scores given mastery.

$$p(score_i|mastery_i)$$

Your "learner" class will be given a constant stream of data from students solving problems. After all observations have completed, the `saveCpt` method will be called. Implement all functions.

```
class Learner():

  # Function: Constructor
  # ---------------------
  # Only if you need it
  def __init___(self):

    self.counter = {}
```

```
# Function: Update
# -----------------
# Called once every time the student completes
# a problem. The score variable is the percent
# correct (0.0 to 1.0) of the student's
# solution. You can discretize this score in any
# reasonable way.
def update(self, score):
   discreteScore = self._getDiscreteScore(score)
   discreteMastery = getMastery()
   if not discreteMastery in self.counter:
     self.counter[discreteMastery] = {}
   counter = self.counter[discreteMastery]
   if not discreteScore in counter:
     counter[discreteScore] = 0
   counter[discreteScore] += 1

# Function: Save Cpt
# ------------------
# To save any data structure you can use
# util.saveProb( yourVariable, fileName).
# Save Cpt is Called once after several days of
# observing students solve triangle inequality
# questions. The variable you save should
# be a valid CPT (not a partial representation)
def saveCpt(self, fileName):
  emissionProb = {}
  for m in self.transCounter:
   masteryTransProb = self._getMasteryEmissionProb(m)
   emissionProb [m] = masteryTransProb
  util.saveProb(emissionProb, fileName)
```

[Helper functions on next page]

```
def _getMasteryEmissionProb (self, m):
    mCounter = self.counter[m]
    mSum = 0.0
    for score in mCounter:
      count = mCounter[score]
      mSum += count
    emissionProb = {}
    for score in mCounter:
      count = mCounter[score]
      prob = float(count) / mSum
      emissionProb[score] = prob
    return emissionProb


def _getDiscreteScore(self, score):
    if score < 0.4: return 'low'
    if score < 0.6: return 'medium'
    return 'high'
```