CS 440 Practice Final Problems
With Answers
Dec 19, 2015

Rules

- Write your answers directly on these pages.
- You may use a single sheet of paper with reference materials (two sides) that you have prepared.
- You may not discuss the test with anyone else or use any electronic devices (the problems involve minimal calculation).
- You may ask clarification questions during the exam. Raise your hand and we will come to you.
- The test has 7 problems. All count equally.
- Show your work for partial credit.

Your name: _____

When you get the exam, you will see something like this. The problems that follow include more than you would see on an actual exam (at least twice as many), and have not been formatted to allow you to write the answers in place. These are real problems. They illustrate the style of the exam—questions that ask you to apply concepts and definitions from class to new problems and cases, and questions that ask you to work out the predictions of algorithms and models from class to specific new examples. This inventory of problems reflects the slightly different emphasis of the course in previous years, so you can expect fewer questions on stuff we didn't spend so much time on and maybe some new questions on stuff that we covered carefully that's not represented here.

1. (Agents) One way to describe the reinforcement learning algorithms we learned is that they compile experience of the world into a program for a reactive agent that acts effectively in that world. Give an example of a world in which a reactive agent cannot act effectively and describe what goes wrong (mathematically and practically) if you try to use reinforcement learning to build an agent for your example.

Reactive agents work as long as they can decide what to do from their current sensory input. However, if the agent sometimes has to make different decisions despite having the same input, it needs to remember clues from past experience that distinguish the two situations. The details can be worked out in many ways. Here's a representative case: the bartender robot. In one state, it gets an order from a customer. In another state, it can get items from the fridge. What it gets out should depend on what the customer has ordered. But (without memory) the robot cannot observe the customer's order when it's at the fridge. So it has no basis to make the decision. Reinforcement learning will learn to pick the drink that maximizes reward. That will be something like the drink that customers order most often in general, not necessarily the drink the customer has ordered in this particular case.

2. (Agents) In discussing face detection, we saw boosting, a machine learning technique that combines simple, inaccurate classifiers into a single, complex, accurate classifier. A classifier created using boosting has similar advantages to the subsumption architecture: both involve simple rules that operate independently and extensibly, with results that are combined together into an effective system. However, the subsumption architecture is designed to choose actions, not make classification decisions. Why don't reinforcement learning algorithms (value or policy iteration or temporal difference or Q-learning) naturally lend themselves to learning a subsumption architecture?

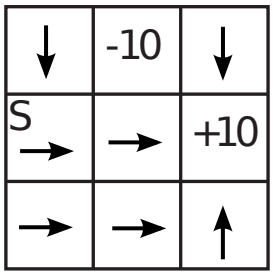
The subsumption architecture is designed to explain behavior as a combination of simple, independent rules for acting. But reinforcement learning algorithms work by ranking all possible choices in light of their overall outcome for the agent and the overall policy that the agent is following. Accordingly, to use reinforcement learning in the subsumption architecture, you'd need to bridge some very deep conceptual gaps. For example, given the utility for the agent, you would need to learn new utility functions that could be used to train individual layers of a subsumption architecture. Then you would need to estimate the utility associated with the decisions of each layer—even though those decisions are interspersed with and even overridden by decisions by other layers. It's not clear how steps like this could work. There are even more problems with thinking of the training in terms of boosting which is based on reweighting a set of training data. Training in reinforcement learning involves acting in the world as the agent learns, so the training data can't be separated from the learning process the way boosting seems to require.

3. (Search) You are given a chemical formula: a mapping from chemical elements to counts. (The domain of the mapping is the set of elements that occur in the molecule. The range is the number of atoms of that element that occur in the molecule. For example, H_2O.) Your job is to find possible molecular structures. To a first approximation, molecular structures can be represented as graphs. Each vertex represents an atom and is labeled with a chemical element and each edge represents a bond and is labeled with an integer (single bond, double bond, etc). In a stable molecule, the sum of the labels on the edges for each vertex must equal the chemical valence of the vertex element. Describe a formal search problem to compute possible molecular structures for a chemical formula. Briefly, what are the states, what are the actions, what are the transitions, what is the initial state, what is a goal? (Your answer can be short English sentences; mathematical definitions are not required. Chemistry could assign a cost to a graph in terms of its energy—but this is beyond the scope of this class.)

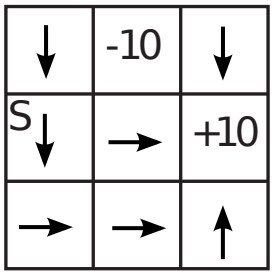
One way to solve this problem is to imagine adding bonds one at a time. That means:

- A state is a labeled graph representing a molecular structure.
- An action is adding a bond (a labeled edge between two vertices). Valid actions must connect two vertices that aren't already connected, and respect the valence limits of those vertices. To make the search practical, you'd need to put some further constraints on actions to avoid creating "the same" structure in multiple ways (and to make sure the graph is connected). For example, one end of the bond should always be the earliest vertex whose valence is not complete; and if the other end is a new vertex, it's enough to consider one vertex from each element.
- A transition adds the corresponding edge to the graph.
- The initial state is an graph with no edges (perhaps with a single vertex distinguished as the "seed" of the search).
- A goal is a (connected) graph with all vertex constraints satisfied.

4. (Markov Decision Processes) You are building a robot to navigate to targets in a 3x3 grid world. The robot knows its position in the grid. It starts at S, gets a reward of 10 units when it reaches a specified square but loses 10 if it winds up in another specified square. After the reward the exercise stops. It costs 1 unit to move each step. The discount is 1 (so future rewards are just as good as present rewards).
(a). Suppose the agent moves with perfect accuracy one square in the direction of its choice. Indicate an optimal policy for the agent below.



(b). Suppose that the agent can try to move one square in the direction of its choice, but the effects of action are unpredictable and the agent has a sizeable chance of slipping. When the agent slips it's equally likely to move to any adjacent square. Indicate an optimal policy for the agent below.



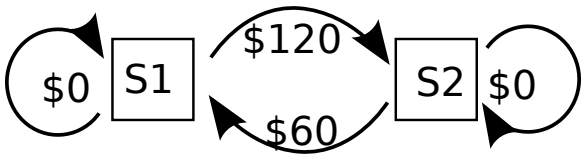
(c). Can these policies be implemented in reactive agents? Explain your answer.

Yes. Markov Decision Processes always lead to reactive policies.

(d). It comes time to actually commit to a design for your robot. Should you implement the first policy, the second policy, or some other strategy? Justify your answer.

You need to measure your agent, to see how accurate its motion is. Once you do that, you can fit the parameters of the Markov Decision Process and calculate the optimal policy.

5. (Markov Decision Processes) Consider a world with two states. When you go from S_1 to S_2 you get \$120. When you go from S_2 to S_1 you get \$60. When you stay you get \$0. Your discount factor is $\frac{2}{3}$. The situation is as shown below:



Suppose you choose an action in S_1 that leads you to S_2 with probability $\frac{1}{3}$ and also choose an action in S_2 that leads you to S_1 with probability $\frac{1}{3}$. Compute the value function (expected utility over the indefinite future) for S_1 and S_2.

Applying the definition of expected utility leads to two equations in two unknowns:

$$V(S_1) = \tfrac{2}{3}(0 + \tfrac{2}{3}V(S_1)) + \tfrac{1}{3}(120 + \tfrac{2}{3}V(S_2))$$
$$V(S_2) = \tfrac{2}{3}(0 + \tfrac{2}{3}V(S_2)) + \tfrac{1}{3}(60 + \tfrac{2}{3}V(S_1))$$

Simplifying by algebra gives:

$$\tfrac{5}{9}V(S_1) = 40 + \tfrac{2}{9}V(S_2)$$
$$\tfrac{5}{9}V(S_2) = 20 + \tfrac{2}{9}V(S_1))$$

Then

$$V(S_1) = 72 + \tfrac{2}{5}V(S_2)$$
$$V(S_2) = 36 + \tfrac{2}{5}V(S_1))$$

Subsituting gives:

$$V(S_1) = 72 + \tfrac{2}{5}(36 + \tfrac{2}{5}V(S_1))$$
$$V(S_1) = (72 * 25 + 10 * 36)/21 = \tfrac{720}{7}$$
$$V(S_2) = 36 + \tfrac{288}{7} = \tfrac{540}{7}$$

Sorry about that last step – I thought I had picked numbers that worked out better. That won't happen on the real test.

6. (Games) A classic illustration of the concepts of game theory is a game known as the prisoner's dilemma. There are two players (A and B). They have two actions: cooperate (C) or defect (D). The payoffs are given below:

		A	
		C	D
B	C	$A:2, B:2$	$A:-1, B:5$
	D	$A:5, B:-1$	$A:0, B:0$

(a). There is one equilibrium. What is it?

> A prefers to play D if the opponent plays C or if the opponent plays D. Same for B. So the equilibrium is that both play D.

(b). Two players can outperform the equilibrium, but only if they make flawed decisions. How?

> If A and B both play C, they get rewards of 2. However, they should both know that they could potentially improve their outcome by changing their decision unilaterally.

(c). Suppose you wanted to use learning and do well both against perfect reasoners and flawed reasoners: what strategy would you choose? For example, do you think you could start out playing optimally and then learn to do better against a flawed player? Or should you start out with a flawed strategy and play optimally if your partner is exploiting you? Hint: what if your opponent is learning the same way you are?

> You would want to start out optimistic, playing C, and switch to D only if your opponent exploits you. This is called "tit for tat". If you start out playing D, then your opponent will see that you are rational and will not try to cooperate with you.

7. (Games) Here is a zero-sum two-player game. The players are A and B. The players each choose actions C and D simultaneously and get the following payoffs:

		A	
		C	D
B	C	$A:2, B:-2$	$A:-2, B:2$
	D	$A:-1, B:1$	$A:3, B:-3$

What is the equilibrium for this game?

A prefers C if B chooses C but D if B chooses D. B has the opposite preference. So both A and B must mix C and D with a probability that balances their outcomes. Suppose A chooses C with probability p. A's choice should leave B indifferent between C and D. Appying B's expected utility to C and D we have:

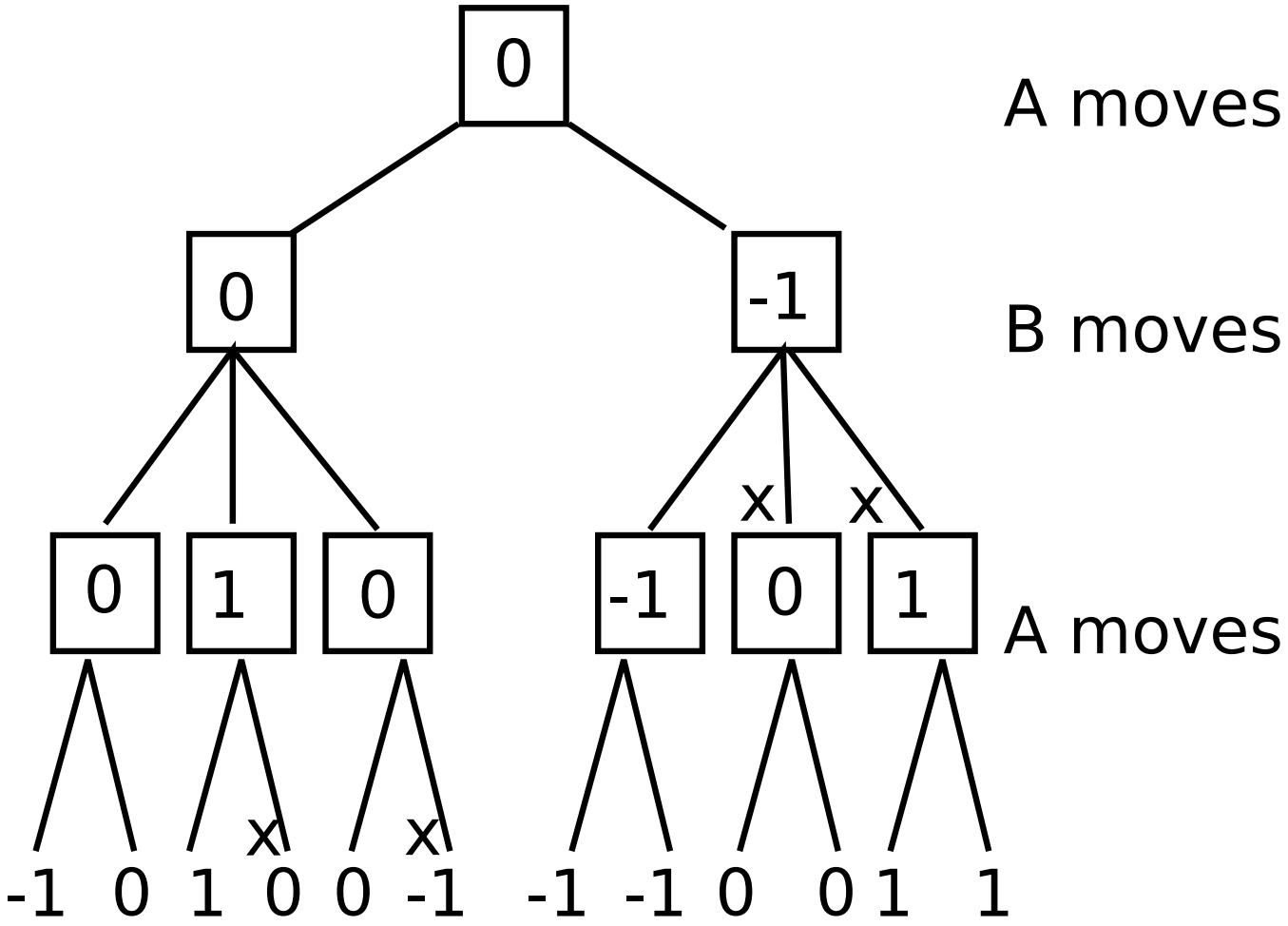
$$-2p + 2(1-p) = p - 3(1-p)$$
$$8p = 5$$
$$p = \tfrac{5}{8}$$

Similar reasoning applies to B, who chooses C with probability q to leave A indifferent:

$$2q - (1-q) = -2q + 3(1-q)$$
$$8q = 4$$
$$q = \tfrac{1}{2}$$

So the equilibrium is: A plays C with probability $\frac{5}{8}$ and D with probability $\frac{3}{8}$; and B plays C with probability $\frac{1}{2}$ and D with probability $\frac{1}{2}$.

8. (Games) The tree below describes a zero-sum turn-taking game with two players, A and B. A moves first, then they alternate. The values at the leaves shows the score for A.



(a). *Write the values for A in each internal node*, assuming both A and B play optimally.
(b). Suppose A is searching for the best strategy, and the search considers nodes from left to right in the order they are diagrammed above. *Write an X next to internal and leaf nodes that don't need to be searched* because they are known not to affect the value of their subtree at the time they are explored.

9. (Learning) A driving robot uses a nearest neighbor model and some training data to predict the maximum speed of a vehicle from a measurement of its size (say, the area in square meters occluded by the car when viewed from behind). You assess the performance of the model by the percentage of the time you get within 10% of the true value. In other words, you want to be very close as often as possible, but it doesn't matter how wrong you are if you're wrong.
You build an initial version of the system and you find many examples in test data where performance is worse considering two neighbors than considering just the closest one. A coworker suggests that if you get enough data, you will eliminate these cases and predicting from two neighbors will work better on all points. Is this necessarily true? Explain.

Sparse data can explain the effect. If the second data point is not very similar to the test point, then it might lead to an inaccurate estimate. However, it is not the only explanation. It may be that the problem is variation in the data set. Then you could have two points that are both close to the test point but lead to different predictions. One prediction will be right, and the other prediction will be wrong, and averaging the two together will decrease performance. This problem will not be resolved by adding more data since it's a feature of the real-world relationship that you are trying to learn.

10. (Learning) Consider an environment with many agents, acting independenty. You want to learn how to act effectively. You can perceive some set of feature values F in your surroundings. These feature values are enough to determine a good reactive policy. You can observe the actions of other agents, and you can also tell what features those agents should perceive from their perspective. Assume that other agents have similar preferences to yours and act effectively in pursuit of them.
(a). Describe a way of using supervised learning to decide what to do, by imitating other agents. What is the exact problem you want to learn, how do you get your training data, how do you represent your training data, and what is your test data?

You want to learn to predict the action of a typical agent given the feature values that they observe in the environment. You watch the agents around you, and record your estimate of their observations in terms of feature values F together with the choice of action that you observe the agent do next. You use some of this as training data and also save some data as test data so you can measure the accuracy of the model. Then you use the learned model to act in your environment. You build a classifier to link feature values F to actions, and then apply the classifier to choose your own action given your perception—in ways that match what other agents around you would typically do.

(b). For (a) we made a bunch of assumptions: others are like you and act well, others have a reactive policy, you can take their perspective. How might supervised learning by imitation fail if these assumptions are violated?

> If agents have different preferences, then you won't be happy acting as they do. If agents agree with you but don't choose wisely, then imitating won't serve you as well as the best choices you could make. If you don't know what agents really see or their choices depend on memories that you can't observe, then you'll make choices in a simpler way than other agents in a way that probably misses out on some of the value that they get.

(c). How would you design an agent to act well in this scenario using reinforcement learning (for example, Q learning)?

> The agent systematically explores actions and their consequences in the environment, and uses that to build a model of actions and their effects, so that you can then choose the action with the best outcome in each possible situation.

(d). Reinforcement learning does not require the assumptions of imitation learning. But what considerations would argue in favor of imitation learning over reinforcement learning when both apply?

> Imitation learning may be faster and less expensive, since you get evidence from many agents acting, not just yourself, you don't have to systematically explore the alternatives, since the agents you observe are already acting well, and you don't have to endure bad outcomes, since you don't actually carry out the actions in training yourself.