Practice Exam

15-381/781: Artificial Intelligence (Fall 2016)

Name:
Andrew ID:

	17	
	17	
	5	
	5	
	5	
	17	
	17	
	17	
Total		

1 Convex Optimization (17 points)

Consider a standard linear program of the form: minimize $\mathbf{c}^T\mathbf{x}$ such that $A\mathbf{x} \leq \mathbf{b}$. Here $\mathbf{x} \in \mathbb{R}^n$ is the vector of variables, and $\mathbf{c} \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $\mathbf{b} \in \mathbb{R}^m$ are constants. Prove from the definitions that this is a convex program.

2 Extensive-Form Games (17 points)

Prove that in Chess, exactly one of the following statements is true:

1. White has a winning strategy, i.e., a strategy such that no matter what Black does, White wins.
2. Black has a winning strategy.
3. Both players can force a tie.

You may use the fact that in an extensive-form game of perfect information, bakward induction gives a subgame perfect Nash equilibrium.

3 Stackelberg Games (5 points)

Suppose two players are playing a *Stackelberg* game with the payoffs given below, with player 1 (the row player) as the leader. What is the payoff of player 1 under the optimal mixed strategy to commit to?

	L	R
U	$(0, 1)$	$(4, 0)$
D	$(-1, 0)$	$(2, 1)$

4 VC-Dimension (5 points)

Prove or disprove: If $\mathcal{C}$ and $\mathcal{C}'$ are two concept classes, then VC-dim($\mathcal{C} \cup \mathcal{C}'$) = VC-dim($\mathcal{C}$)+VC-dim($\mathcal{C}$).

5 Uninformed Search (5 points)

Consider the problem of uninformed search on a *finite* search space with a binary tree structure as shown in the following figure (the actual space could be larger).

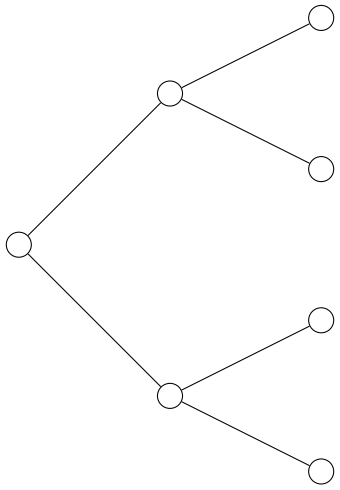

Figure 1: The structure of the search space

Suppose there are multiple goal states, we want the shortest path to the closest goal, and we want to minimize time. We have unlimited memory. Discuss which of BFS, DFS, or IDS would be best suited and why. Briefly explain why the other options are not as well suited as your choice.

6 Informed Search (17 points)

The 8-Puzzle game is a single-player board game which consists of a 3×3 board with 8 tiles and 1 blank slot. A tile can move horizontally or vertically to its adjacent blank slot (if it neighbors it). The objective of the game is to start from a given arrangement of tiles and move tiles to achieve a given goal arrangement.

Here, we discuss some heuristics to be used with A* graph search. Recall, from slides 18 and 19 of Lecture 2, that heuristic $h_1(\cdot)$ returns the number of tiles that are in the wrong position and $h_2(\cdot)$ returns the sum of Manhattan distances of tiles from their goal positions. We introduce a third heuristic, $h_3(\cdot)$, that is the *minimum* number of moves necessary to get to the goal state if each action could move any tile to the blank slot. This is another *relaxed problem* heuristic.

1. (9 points) Prove that h_3 is consistent.
2. (4 point) Prove or disprove: h_3 dominates h_1.
3. (4 point) Prove or disprove: h_3 dominates h_2.

7 Deep Learning and Computer Vision (17 points)

1. (8 points) Consider a layer in a convolutional neural network that takes in one 100×100 feature map (e.g., a gray-scale image), and outputs 100 feature maps. In each of the following cases, give the number of parameters that must be learned for this layer. Remember that we include a bias value for each output map.

   (a) The layer is a convolution layer where the filters are the same size as the input feature map.

   (b) The layer is a convolution layer with 10×10 filters and a stride of 5.

   (c) The layer is a locally-connected layer with 10×10 tiles and a stride of 5.

   (d) The layer is a convolution layer with 1×1 filters and a stride of 1.

2. (5 points) What is the trade-off with having more or fewer parameters?

3. (4 points) Explain an additional reason (besides the number of parameters) convolution layers with filters that are smaller than the input map but larger than 1×1 are well-suited for image-related tasks.

8 (From Homework) Choosing the Value of $R_{\max}$ (17 points)

This question refers to the R-MAX algorithm (to be covered in lecture the week of 10/10).

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

8.1 Setting an upper bound [3 points]

Suppose we have a known upper bound for the reward and we set $R_{\max}$ to be this upper bound. When this bound is very loose, i.e., the bound is much greater than the true maximum possible reward, how will the algorithm behave?

8.2 Working up to the true bound [8 points]

Instead, suppose we initialize $R_{\max} = 0$ (assume this 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 our unknown states accordingly). While intuitively this may seem like it should work, in some cases it does not. Give a simple MDP where this fails, i.e., where we will never find the optimal policy. Show that we never find the optimal policy on your example under balanced wandering.

8.3 Optimistically increasing $R_{\max}$ (6 points)

Now suppose we use the same algorithm as in (1.1.2), but instead we set $R_{\max} = \rho + \delta$, for some δ, which may depend on the discount factor, γ. Either give a brief description of why this won't fail as it did in (1.2.2), or show that for any value of δ, you can construct an MDP where this fails.