STANFORD UNIVERSITY
CS 221 Midterm, Fall 2007

Question	Points
1 Short Answers	/18
2 Motion Planning	/12
3 Search Space Formulation	/14
4 A*	/12
5 Supervised Learning	/20
6 Markov Decision Processes	/16
7 Computer Vision	/8
Total	/100

Name of Student: ______________________________

Exam policy: This exam is open-book and open-notes. Any printed material that you brought with you is allowed. However, the use of mobile devices is not permitted. This includes laptops, cellular phones and pagers.

Time: 3 hours.

Length: This midterm contains 21 pages. The last two pages are blank pages that you can use if you need extra space to answer some question.

The Stanford University Honor Code:

I attest that I have not given or received aid in this examination, and that I have done my share and taken an active part in seeing to it that others as well as myself uphold the spirit and letter of the Honor Code.

Signed: ______________________________

1. **Short answers [18 points]**

   The following questions require a true/false accompanied by one sentence of explanation, or a very short answer (also accompanied by a brief explanation).

   To discourage random guessing, one point will be deducted for a wrong answer on multiple choice (such as yes/no or true/false) questions! Also, no credit will be given for answers without a correct explanation.

   (a) [**3 points**] In class, we noted that grid-based discretization for motion planning works well in 2-4 dimensional problems, and studied probabilistic roadmaps for higher dimensions. However, since we live in a 3-dimensional world, most real motion planning problems in robotics can be solved in a reasonable about of time using grid-based discretization. [**True/False**]

   (b) [**3 points**] Suppose h is an admissible heuristic for a search problem, such that $h' = 2h$ is *not* admissible. Then A* search with the heuristic function h' will never expand more nodes than A* search with the heuristic function h. [**True/False**]

   (c) [**3 points**] Suppose we are interested in finding *all* solutions to a constraint satisfaction problem. Say, for an 8-queens problem, instead of asking for any one solution (i.e., any one arrangement in which the 8 queens lie on different rows, columns and diagonals), we want all possible solutions (i.e., all such arrangements).

   Which of the following techniques would still be useful for constructing efficient algorithms for finding all solutions?

   i. Forward checking.
   ii. Choosing the next value to assign a variable using the least constraining value heuristic.
   iii. Choosing the next variable to instantiate using the minimum remaining values heuristic.

(d) **[3 points]** An MDP has a reward function R, optimal value function V^* and optimal policy π^*. Consider new reward functions:

i. $R_1(s) = R(s) + 10$.

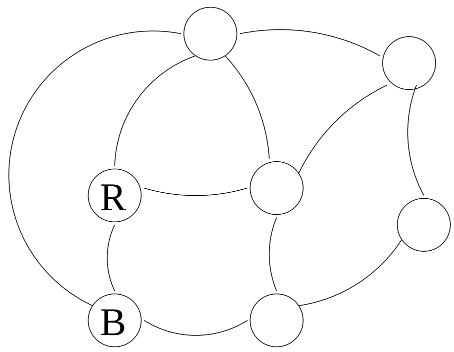
ii. A reward function R_2 such that whenever $R(s_1) > R(s_2)$ for two states s_1 and s_2, then we also have $R_2(s_1) > R_2(s_2)$. (You can assume that the reward $R(s)$ is different for each state s.)

Consider replacing the reward function R in the original MDP by each of these reward functions. In which of the above two cases (if any) must π^* still be an optimal policy in the new MDP?

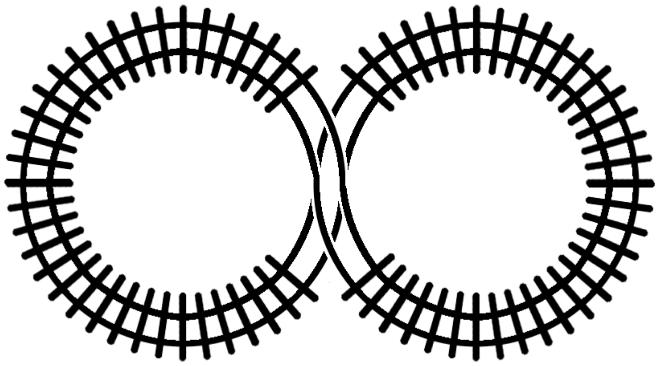
(e) **[3 points]** Consider a pair of points P_1 and P_2 in physical space. Let their corresponding projections onto the image plane in a perspective camera be p_1 and p_2 respectively. Then the four points P_1, P_2, p_1 and p_2 always lie in a plane. **[True/False]**

(f) **[3 points]** Consider the graph coloring constraint satisfaction problem on the next page. In this problem, each circle denotes a variable with domain R,B,G, and an edge between two circles denotes the binary constraint that the corresponding variables must be assigned different values.

Suppose the two leftmost variables have already been assigned the values R and B, as shown in the figure. Show the result of applying arc consistency checking to this instance. You should write the resulting domain of each variable next to the corresponding circle.



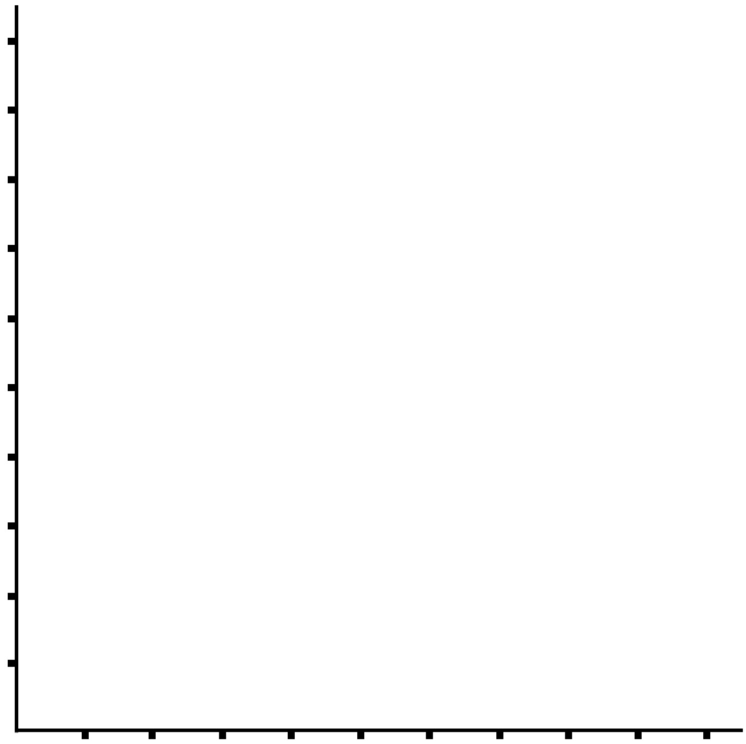
2. **Configuration Spaces [12 points]**



Suppose you want to run two robotic model trains on a small figure-eight track. Each train can move forward and backwards along the track. No branching is allowed at the intersection in the middle, i.e., a train entering the intersection from the bottom-right direction may exit the intersection only through the upper left. Each train is of length l. Each half of the track is a circle of circumference $5l$. (Thus, the total length of track in the above figure is $10l$.) For simplicity, you may assume that the trains and tracks have 0 width, and that the trains always fit the curve of the track perfectly.

(a) **[3 points]** Give a representation of the configuration space for the train robots. State precisely how your variables correspond to the workspace, including which part of each train you use as the reference point.

(b) **[9 points]** Using your configuration space formulation from part (a), draw the obstacles in your configuration space on the axes provided on the next page. Clearly label what each axis corresponds to. Provide algebraic expressions for the locations of any key points on these obstacles.



3. **Search space formulation [14 points]**

Dr. Aye Starr heads the NASA team of N engineers that will land the next robot on planet Mars. Once the robot lands on Mars, it will be operational for $T = 5N$ days. Each day, it will be remote-controlled from Earth via satellite by one of N NASA engineers. Controlling the robot is a taxing, mission-critical job, and Dr. Starr wants to reduce costs, as well as reduce the chances of error.

Dr. Starr asks for your help in planning the assignment of engineers to each of the T days. Here are the basic requirements for an assignment:

- Exactly one of the N engineers must be assigned on each of the T days.
- No engineer can be assigned on more than 10 days.

(a) **[8 points]** The engineers are extremely picky, and demand different amounts of money for being assigned on different days. Suppose NASA has to pay $c(n, t)$ dollars to assign engineer n on day t (where $1 \leq n \leq N$, $1 \leq t \leq T$ and $c(n, t) > 0$). The task is to find the assignment that minimizes the total amount of money that NASA has to spend. Formulate the task as a search problem.

You should provide a precise description of the components in your formulation (state space, initial state, operators, goal test), of the constraints under which each operator can be applied, and of the effects of each operator on the state components. You may use either English or pseudocode.

Make sure that your formulation of the search space has no problem with repeated states; i.e., make sure that no state can be reached from the initial state by two different paths.

(b) **[6 points]** Now consider the scenario where NASA only cares about the success of the mission, and doesn't care how much money it has to pay.

Specifically, NASA figures out that if engineer n is assigned on day t (where $1 \leq n \leq N$ and $1 \leq t \leq T$), the probability of a critical error is given by $e(n, t)$ (where $0 < e(n, t) < 1$), *independent of all the other days.* If such a critical error is made on even one of the T days, the whole mission has to be aborted and is unsuccessful. The mission is successful if and only if *no* critical errors are made on any of the T days.

How can we modify the formulation in part (a) to find the assignment that minimizes the probability of an unsuccessful mission (instead of minimizing the cost in dollars of assigning engineers)? Be sure to precisely describe any changes that need to be made to your answer to part (a). You do not have to give a heuristic funcion.

(Hint: Write down the probability of an unsuccessful mission given a complete assignment.)

4. **A* search [12 points]**

   Consider applying A* with an *admissible* heuristic h that is guaranteed to be "useful" in the following sense: for any node n, we know that $h(n) \geq h^*(n)/2$. As usual, let the optimal path cost to a goal node be c^*, and assume that all edge costs are positive.

   Prove that if A* search with the heuristic h expands a node n_1, then this node must lie on a path from the initial state to some goal node with total path cost at most $2c^*$. (Or equivalently, prove that A* search never expands a node n_2 such that all paths from the initial state to a goal node via node n_2 have path cost greater than $2c^*$.)

5. **Supervised Learning [20 points]**

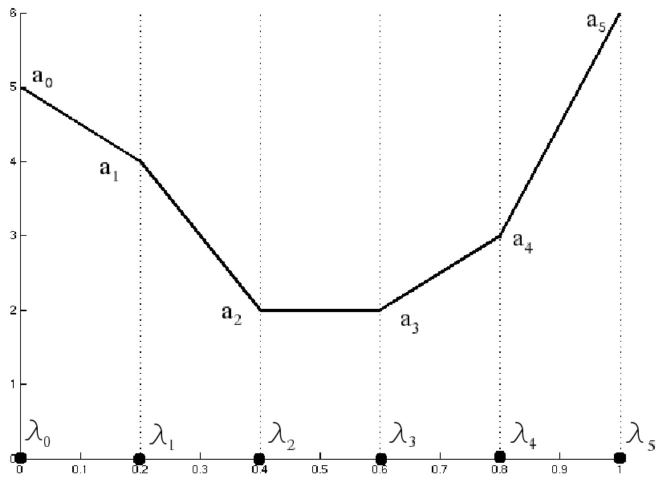
Suppose we are trying to predict a real-valued target variable y as a function of a real-valued input variable x, where $0 \leq x \leq 1$. In class, we studied how a polynomial function can be fit to data by least squares regression. In this problem, we will consider how to fit a different class of functions, namely **piecewise linear** functions.

Specifically, let $N \geq 2$ be a fixed integer that represents the number of linear segments in our function. We take $N+1$ evenly spaced grid points $\lambda_0 = \frac{0}{N}, \lambda_1 = \frac{1}{N}, \ldots, \lambda_N = \frac{N}{N}$. The function is defined by choosing a value $a_j = f(\lambda_j)$ for each of the grid points λ_j. For $0 \leq x \leq 1$, we define:

$$f(x) \quad = \quad \begin{cases} a_j & \text{if } x = \lambda_j \text{ for some } j \\ (1-\alpha)a_j + \alpha a_{j+1} & \text{otherwise, where } \lambda_j < x < \lambda_{j+1} \text{ and } \alpha = \frac{x-\lambda_j}{\lambda_{j+1}-\lambda_j} \end{cases}$$

Note: The second line in the formula above is just performing linear interpolation between $x = \lambda_j$ and $x = \lambda_{j+1}$. You should not need the exact equation for the rest of this problem.

An example function with $N = 5$ segments is shown in the following figure:



Given a training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, our learning algorithm will pick the parameters a_j that minimize the the squared error criterion:

$$J(a_0, \ldots, a_N) = \frac{1}{2}\sum_{i=1}^{m}(f(x^{(i)}) - y^{(i)})^2.$$

(a) **[4 points]** Suppose you fit a piecewise linear function to a given training set, and you then discover that your model matches the training set perfectly, but gives high test error. Should you increase or decrease N, the number of segments? Why?

For the rest of this problem, we consider the simpler case of $N = 2$ segments.

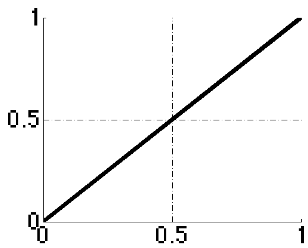
We will minimize the squared error criterion $J(a_0, \ldots, a_N)$ by converting this problem into a standard least-squares linear regression problem parameterized by a vector of parameters θ, using the following features:

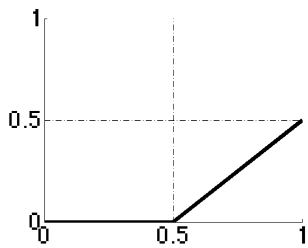
$$\phi(x) = \begin{pmatrix} \phi_0(x) \\ \phi_1(x) \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ \max(x - 0.5, 0) \\ 1 \end{pmatrix}.$$

The notation $\max(x - 0.5, 0)$ denotes the function

$$\max(x - 0.5, 0) = \begin{cases} x - 0.5 & \text{if } x \geq 0.5 \\ 0 & \text{if } x < 0.5 \end{cases}$$

The functions ϕ_0 and ϕ_1 are plotted below:





(b) **[6 points]** Suppose that a_0, a_1 and a_2 are given (corresponding to $f(0)$, $f(0.5)$ and $f(1)$ respectively).
Show that we can use the given a_0, a_1 and a_2 values to construct a vector $\theta = (\theta_0, \theta_1, \theta_2)^T \in \mathbb{R}^3$ such that our piecewise linear function f can be represented as a linear function of the features: $f(x) = \theta^T \phi(x)$ for $0 \leq x \leq 1$. Derive the equation for θ in terms of the a_0, a_1 and a_2 values.

(**Hint:** To derive this equation, it is sufficient to consider the following conditions on the function values at the grid points: $\theta^T \phi(0) = a_0$, $\theta^T \phi(0.5) = a_1$, and $\theta^T \phi(1) = a_2$.)

We want to enforce the condition that $f(x)$ be fairly similar for nearby values of x. Therefore, in addition to the least-squares error penalty, we also introduce the following term into our cost function:

$$\Omega(a_0, \ldots, a_N) = \frac{1}{2}\sum_{j=0}^{N-1}(a_{j+1} - a_j)^2$$

We are now trying to minimize the function

$$\frac{1}{2}\sum_{i=1}^{m}(f(x^{(i)}) - y^{(i)})^2 + \frac{1}{2}\sum_{j=0}^{N-1}(a_{j+1} - a_j)^2.$$

(c) **[4 points]** Explicitly express the cost function given above as a function of θ, $\phi(x^{(i)})$ and $y^{(i)}$ for the case where $N = 2$. In particular, show that the cost function can be written as:

$$\frac{1}{2}\sum_{i=1}^{m}(\theta^T\phi(x^{(i)}) - y^{(i)})^2 + \frac{1}{8}\theta_0^2 + \frac{1}{8}(\theta_0 + \theta_1)^2$$

(d) **[6 points]** Derive the batch gradient descent update rule for θ_0 (i.e., only the weight corresponding to ϕ_0) using the new cost function given in part (c), still with $N = 2$.

(e) **[2 extra credit points]** Name one reason it may be advantageous to include $\Omega(a_0, \ldots, a_N)$ in the cost function. (You don't need to limit yourself to the case where $N = 2$.)

6. **Markov Decision Processes [16 points]**

We have an MDP with reward function $R(s)$, transition probabilities $P_{sa}(s')$, and discount factor $0 \leq \gamma < 1$. We are also given a biased coin that lands Tails with probability α and Heads with probability $(1-\alpha)$, where α is known.

Suppose we have a policy π that behaves in the following way: for a given state s, the policy first tosses the biased coin. If the coin lands Heads, it executes the optimal policy $\pi^\star$ (i.e., chooses action $\pi^\star(s)$); if the coin lands Tails, it executes some other fixed policy $\hat{\pi}$. In this problem, we will prove a bound on difference between V^π (the value function for π) and the optimal value function V^{π^*} (which is also written V^*).

Note: You can also get full credit on this problem by proving just part (d), as long as you do not use the results from parts (a)-(c). If you use the results from those parts but do not prove them, you will only get credit for part (d).

(a) **[4 points]** Express the transition probability $P_{s\pi(s)}(s')$ in terms of α, $P_{s\pi^\star(s)}(s')$ and $P_{s\hat{\pi}(s)}(s')$.

(b) **[4 points]** For any state s, prove the following statement relating the value function V^π for policy π with the value function $V^{\pi^\star}$ for the optimal policy $\pi^\star$:

$$V^\pi(s) - V^{\pi^\star}(s) \quad = \quad \gamma \sum_{s' \in S} \left[P_{s\pi(s)}(s') \left(V^\pi(s') - V^{\pi^\star}(s') \right) + \left(P_{s\pi(s)}(s') - P_{s\pi^\star(s)}(s') \right) V^{\pi^\star}(s') \right]$$

Hint: Recall Bellman's equations for V^π and $V^{\pi^\star}$:

$$V^\pi(s) \quad = \quad R(s) + \gamma \sum_{s' \in S} \left[P_{s\pi(s)}(s') V^\pi(s') \right], \quad V^{\pi^\star}(s) \quad = \quad R(s) + \gamma \sum_{s' \in S} \left[P_{s\pi^\star(s)}(s') V^{\pi^\star}(s') \right]$$

(c) **[4 points]** Using the result of 6a and 6b, prove the following for any state s:

$$\left|V^{\pi}(s) - V^{\pi^\star}(s)\right| \leq \gamma \sum_{s' \in S} \left[P_{s\pi(s)}(s') \left|V^{\pi}(s') - V^{\pi^\star}(s')\right| + \alpha \left|P_{s\hat{\pi}(s)}(s') - P_{s\pi^\star(s)}(s')\right| \cdot \left|V^{\pi^\star}(s')\right|\right]$$

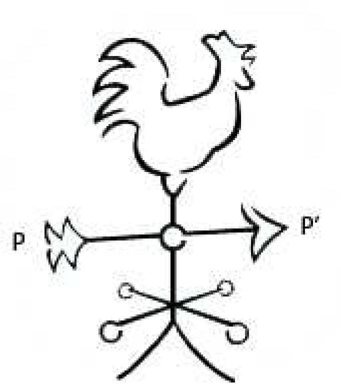
(d) **[4 points]** Recall that the "infinity norm" of a value function V is defined as:

$$||V||_\infty = \max_s |V(s)|.$$

Without further assumptions, prove the following using the result of 6c:

$$||V^{\pi} - V^{\pi^\star}||_\infty \leq \frac{2\gamma\alpha}{1-\gamma} ||V^{\pi^\star}||_\infty$$

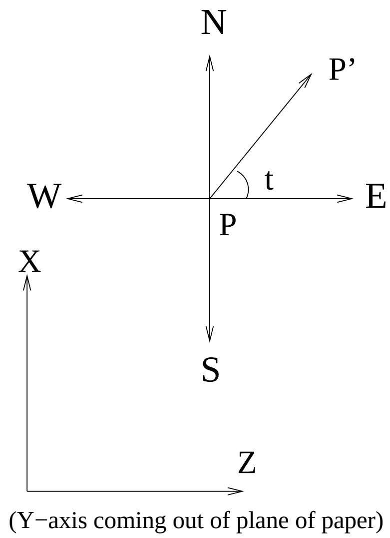
7. **Computer Vision [8 points]**



Your kite-flying robot needs to know which direction the wind is blowing and has fortuitously found a weathervane with its arrow at known height Y. The robot identifies a point P at the back of the arrow and a point P' at the front of the arrow, both at height Y. Their coordinates as projected onto the robot's image plane are (x, y) and (x', y'), respectively. The robot is facing due east with the Z-axis of its camera perfectly horizontal and at height 0. The X-axis is pointing due north. The focal length of the camera is known to be f. The length of the vane is unknown.

In what direction is the wind blowing? (Assume that the wind is blowing in the direction from P to P'.) Report your answer as an expression for the angle between due east and the wind. I.e., report the angle t as pictured in the diagram below, as a function of x, y, x', y', Y and f.

(**Hint:** Because the height Y of the vane is known, you should be able to find the 3-D coordinates of points P and P'.)



(Space for previous question.)

(Extra space for answers)

(Extra space for answers)