CS 4758/6758: Robot Learning: Homework 3
Due: Mar 10 (Thursday), 5pm

1 Short Questions (15 pts)

1.1 Potential Fields (6 pts)

Each of the scenarios below represents a potential field.

The attractive component of the field points towards the circled dot, with magnitude:

$$k_{att}\left((x - x_{goal})^2 + (y - y_{goal})^2\right)$$

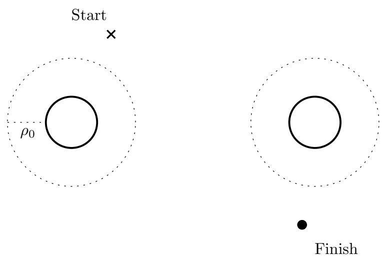
There are two repulsive components, each pointing away from one of the circular walls, with magnitude:

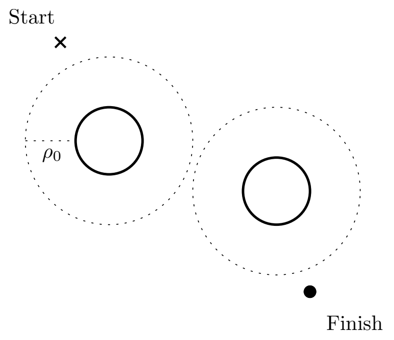
$$\begin{cases} k_{rep}\left(\frac{1}{\rho(x,y)} - \frac{1}{\rho_0}\right)^2 & \rho(x,y) \le \rho_0 \\ 0 & else \end{cases} \quad \text{(where } \rho(x,y) \text{ is the distance to the wall)}$$

Assume $k_{rep} >> k_{att}$.

Questions:

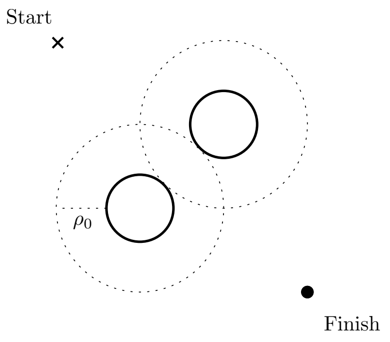
(a) For each of the two cases below, draw the path taken from the starting position to the goal.





(b) In the scenario below, this method may not work. One way to make it work is to change ρ_0. Would you increase it or decrease it to make it work? Justify in a line.

Is there some other potential you could add to make it work? Explain briefly.



1.2 Filters (3 pts)

Often we have to apply multiple filters in series. I.e., apply one filter, followed by applying another filter on the output of the first filter.

We have three filters: Gaussian smoothing filter G(x), Sobel edge filter S(x) and median filter M(x). Please mark which of the following is *true* or *false*:

1. G(S(x)) = S(G(x))
2. M(G(x)) = G(M(x))
3. S(G(M(x))) = G(S(M(x)))

1.3 Linear Systems (6 pts)

1. We define the state of a robot (e.g., for a helicopter to stabilize its height) by $x = (h, \dot{h})$, where h is the height of the helicopter and $\dot{h}$ would be the velocity in vertical direction (rate of change of height). We model it as a *continuous* time linear system $\dot{x} = Ax$, where $A \in \Re^{2\times2}$ would describe the linear system.[1]

   We bought two quadrotors, and based on measurements we found out that their effective dynamics matrix is given as follows.

$$A_1 = \begin{pmatrix} 0 & 1 \\ 0.1 & 1 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 0 & 1 \\ -0.1 & -1 \end{pmatrix}$$

   Please determine explain which helicopter(s) will be passively stable out of the box.

2. Now you were hired to comment on a autonomous flying plane robot a company is building. The plane's state in this case is determined by $x = (u, v, \theta, q)$, where: u is the velocity of aircraft along body axis; v is the velocity of aircraft perpendicular to body axis (down is positive); θ is the angle between body axis and horizontal (up is positive); q is the angular velocity of aircraft (pitch rate).

   The plane engineers told you that the continuous time linearized dynamics matrix of the plane flying at 774ft in this case is given by:

$$A_1 = \begin{pmatrix} 0.003 & 0.039 & 0 & 0.322 \\ 0.065 & 0.319 & 7.74 & 0 \\ 0.020 & 0.101 & 0.429 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

   Please explain if their plane is stable or not?

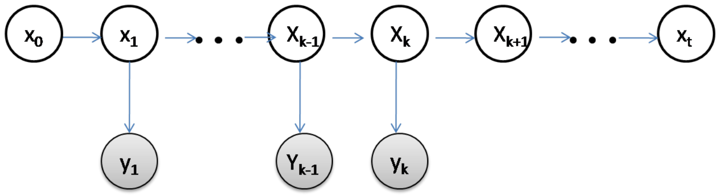
   Hint: In a continuous time linear system, eigenvalues of the system matrix can tell us about stability. Note that the conditions are different for discrete time linear systems.

[1] In the class, we have looked at both continuous time and discrete time cases. In discrete time (e.g., Kalman filters), we have $x(t+1) = Ax(t)$.

2 Hidden Markov Models: Theory (25 pts)

For a hidden Markov process, we discussed in class how to calculate $\hat{\mathbf{x}}_{t|t}$—the most likely state at time t, given observations $\mathbf{y}_1, \mathbf{y}_2, \ldots \mathbf{y}_t$ for all times $\leq t$.

In this question, we are interested in estimating $\hat{\mathbf{x}}_{t|k}$ where $k < t$. Thus some of the states (specifically, for times $k+1$ to t) do not have corresponding observation. Intuitively, this means trying to predict the future position of a robot. In this problem, you will derive algorithms for computing $\hat{\mathbf{x}}_{t|k}$



An HMM is fully described by the following **"terms"**:
(a) prior state $\Pr(\mathbf{x}_0)$,
(b) transition probabilities $\Pr(\mathbf{x}_t|\mathbf{x}_{t-1})$, and
(c) emission probabilities $\Pr(\mathbf{y}_t|\mathbf{x}_t)$.

We want to devise a recursive expression for $\Pr(\mathbf{x}_t, \mathbf{y}_1 ... \mathbf{y}_k)$ in terms of these terms, so that it can be efficiently computed.

Questions:

1. Write an expression for $\Pr(\mathbf{x}_k|\mathbf{y}_1 ... \mathbf{y}_k)$ in terms of the HMM "terms". Show your derivation.
2. Write an expression for $\Pr(\mathbf{x}_t|\mathbf{y}_1 ... \mathbf{y}_k)$ in terms of the HMM "terms". Show your derivation.

3 Hidden Markov Model: Programming (60 points)

The first part of the assignment is to build an HMM from data. Recall that an HMM involves hidden state that changes over time, as well as observable evidence, henceforth called the output of the HMM.

Regarding the distribution over the start state (P(X[0])): In this assignment, assume that there is a single dummy start state, distinct from all other states, and to which the HMM can never return. Even so, you will need to estimate the probability of making a transition from this dummy start state to each of the other states. [2]

CS4758: *During this training phase, we assume that the state variables are visible.* Given these sequences, you need to estimate the probabilities that define the HMM. (Hint: You will not need to do Expectation Maximization (EM).)

CS6758: *During this training phase, the state variables are NOT visible.* Given these sequences, you need to estimate the probabilities that define the HMM. (Hint: You will need to do Expectation Maximization (EM).)

The second part of the assignment is to write code that computes the most probable sequence of states (according to the HMM that you built from data) for a given sequence of outputs. This is essentially the problem of implementing the Viterbi algorithm as described in class.

Question.

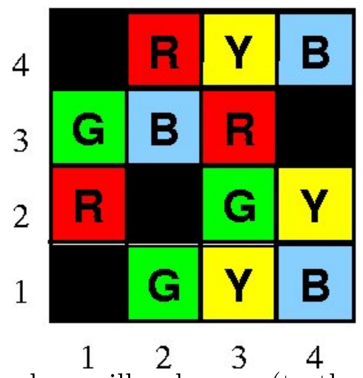
A robot is wandering through the following small world, as shown on the figure.

[2]Although this approach corresponds to the meaning of a conditional probability, when making estimates of this sort, it is often preferable to smooth the estimates.

The robot can only occupy the colored squares. At each time step, the robot attempts to move up, down, left or right, where the choice of direction is made at random. If the robot attempts to move onto a black square, or to leave the confines of its world, its action has no effect and it does not move at all. The robot can only sense the color of the square it occupies. However, its sensors are only 90% accurate, meaning that 10% of the time, it perceives a random color rather than the true color of the currently occupied square. The robot begins each walk in a randomly chosen colored square. (These numbers are only approximate, you'd need to estimate them from the training set for the homework.

In this problem, state refers to the location of the robot in the world in x:y coordinates, and output refers to a perceived color (r, g, b or y). Thus, a typical random walk looks like this:

3:3 r
3:3 r
3:4 y
2:4 b
3:4 y
3:3 r
2:3 b
1:3 g
2:3 b
2:4 r
3:4 y
4:4 y



Here, the robot begins in square 3:3 perceiving red, attempts to make an illegal move (to the right), so stays in 3:3, still perceiving red. On the next step, the robot moves up to 3:4 perceiving yellow, then left to 2:4 perceiving blue (erroneously), and so on.

By running your program on this problem, you will build an HMM model of this world. Then, given only sensor information (i.e., a sequence of colors), your program will re-construct an estimate of the actual path taken by the robot through its world.

The data for this problem is in `robot_no_momentum.data`, a file containing 200 training sequences (random walks) and 200 test sequences, each sequence consisting of 200 steps.

We also are providing data on a variant of this problem in which the robot's actions have "momentum" meaning that, at each time step, with 85% probability, the robot continues to move in the direction of the last move. So, if the robot moved (successfully) to the left on the last move, then with 85% probability, it will again attempt to move left. If the robot's last action was unsuccessful, then the robot reverts to choosing an action at random. Data for this problem is in `robot_with_momentum.data`.

What to report? For both cases: (a) Submit your code. (b) Submit printout of your learned matrices from the data. (c) Average errors the predictions made. (d) Submit the ground-truth and the predicted trajectory from your HMM for trajectory first and the second sequence in the data.

Homework question rules:

You cannot use an existing implementation of HMM. You have to implement your own and you can use any programming language.