CS 4758/6758 Robot Learning: Homework 4

Due April 11 in class

1 Potential Fields (12 points)

Each of the scenarios below represents a potential field.
The attractive component of the field points towards the circled dot, with magnitude:

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

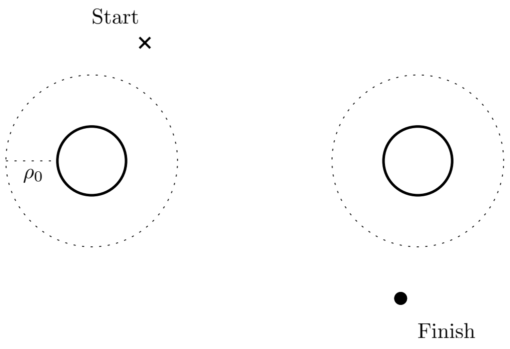
There are two repulsive components, each pointing away from one of the circular walls, with magnitude:

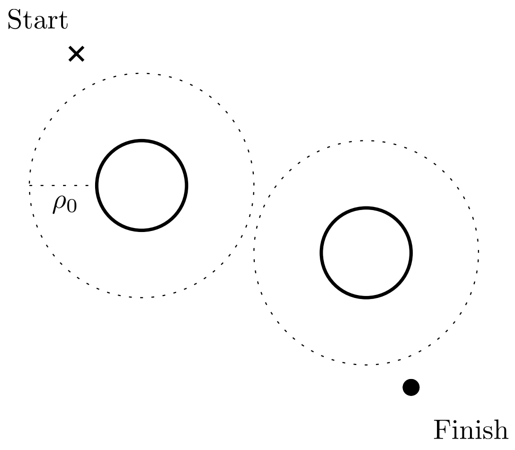
$$\begin{cases} k_{rep}\left(\frac{1}{\rho(x,y)} - \frac{1}{\rho_0}\right)^2 & \rho(x,y) \leq \rho_0 \\ 0 & else \end{cases} \quad \text{(where } \rho(x,y) \text{ is the distance to the wall)}$$

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

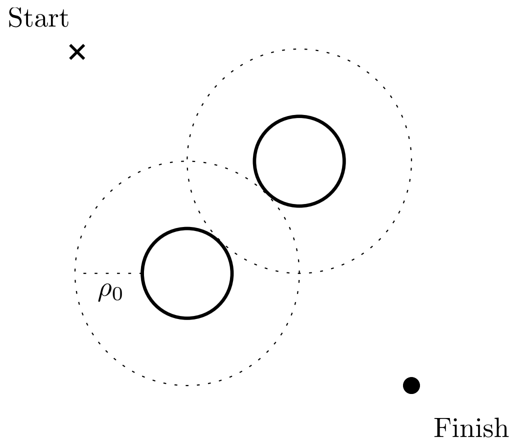
Questions:

(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.



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

2 Proportional Control (8 points)

Consider the linear system

$$x(t+1) = \begin{bmatrix} 10 & 1 \\ 1 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 1 & .1 \\ .1 & 1 \end{bmatrix} u(t)$$

where $u(t)$ is a control signal:

$$u(t) = \begin{bmatrix} K_{p1} & 0 \\ 0 & K_{p2} \end{bmatrix} (x_{desired} - x(t))$$

For each of the following cases, say whether or not the resulting system is stable. Show your work. (4 points each)

(a) $K_{p1} = 1$, $K_{p2} = 1$

(b) $K_{p1} = 10.5$, $K_{p2} = .5$

3 Particle Filters (10 pts.)

Consider the following proposition:
There is no point in using a particle filter if the number of distinct states at any time is no more than the number of particles one plans to allocate to the particle filter.

Justify or refute this statement.

4 HMM Programming (50 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.

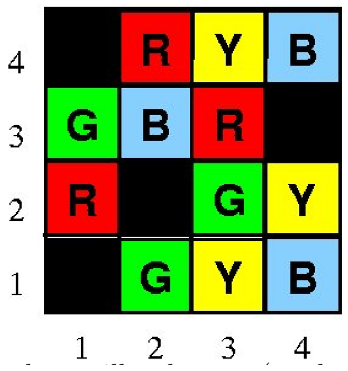
During this training phase, the state variables are NOT visible. Given these sequences, you need to estimate the probabilities that define the HMM. (Hint: You will need to do Expectation Maximization (EM).)

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.

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 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 se- quences (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.

5 True/false (20 points)

For each statement, please state whether it is true or false and justify or refute in a line:

1. In linear regression, adding features will never increase training error.
2. In policy iteration, the complexity of each step (iteration of the policy update loop) is linear with the number of states.
3. The problem of finding the most probable path in an HMM always gives an optimal answer with respect to the current model.
4. For a Kalman filter, given the model $\{A, B, H, Q, R\}$ and a known initial state, we can obtain perfect knowledge of uncertainties in prediction at all future times regardless of what is observed.
5. Local linearity in the state transitions is the only requirement for the Extended Kalman Filter to work reasonably well