Project: Robot Localization

Background and Setup of the Problem

A robot performs a random walk in a room, whose floor has been been marked into an $N \times N$ grid for the purpose of identifying the position of the robot at any time. Therefore, the robot's position can be given as (x, y) where $x = 0, 1, ..N-1$ and $y = 0, 1, ..N-1$. The robot moves either towards right, left, up or down on the grid, with equal probability of 1/5 or, with probability 1/5 it chooses to stay at its position. At the edges and corners of the grid, since the robot does not have all the five options, the probability of staying at its position increases by 1/5 for each restricted dimension. (For example, at (0,0), where it cannot move up and left, the robot has probability 3/5 to stay at the same position. At (1,0) the probability of staying at its position is 2/5.) The robot makes a move (that also includes the decision to stay there) in discrete steps of time, i.e., after every t seconds. Fig. 1 shows the sketch of robot's random walk.

Suppose you place the robot at a known location, for instance (0,0), in the room and leave it for some time, during which the robot completes the random walk for K steps and then it stops. During its walk, after every step, the robot obtains a ranging signal from the four beacons as shown in Fig. 2. (A beacon is a fixed transmitting device that sends a ranging signal according to a pre-determined format to help localize a moving object, like robot in this case.) From these ranging signals, the robot can infer its position. For instance, the robot can estimate the distance to each of the beacons from the ranging signal. Since the robot knows the locations of the beacons, he can then create an observation model for its own position. We assume the

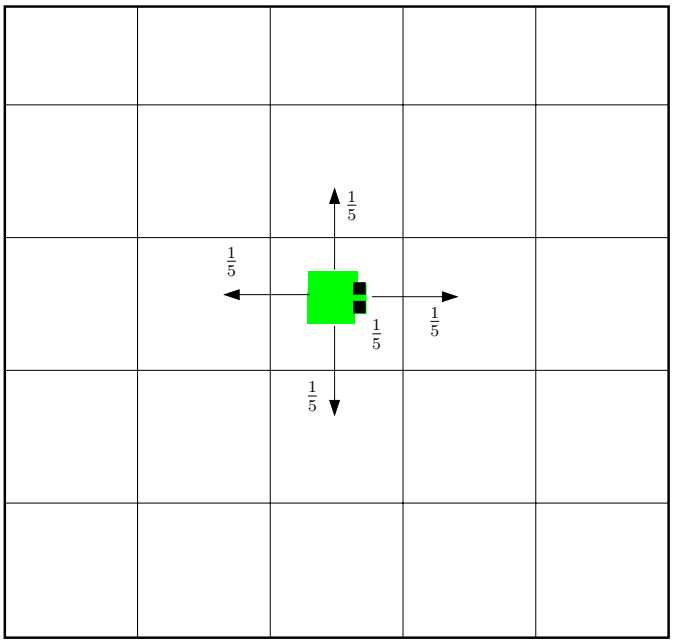

Figure 1: Robot performing random walk on an $N \times N$ grid

following model for the estimated position:

$$\mathbf{r}_k = \mathbf{s}_k + \mathbf{n}_k, \qquad k = 0, 1, ...K \tag{1}$$

where $\mathbf{s}_k$ is the actual position vector of the robot at the kth step, $\mathbf{r}_k$ is the calculated position vector and $\mathbf{n}_k$ is the additive noise component at the kth step. All vectors have two components; for example, $\mathbf{s}_k = [x_k, y_k]$, where x_k and y_k are the x-position and y-position of the robot on the grid. Similarly, $\mathbf{n}_k = [n_{x_k}, n_{y_k}]$. Assume that n_{x_k} and n_{y_k}, for all k, are independent samples from $\mathcal{N}(0, \sigma^2)$.

These estimated positions are stored in the form of a sequence in the robot and you read this sequence from its memory after you come back.

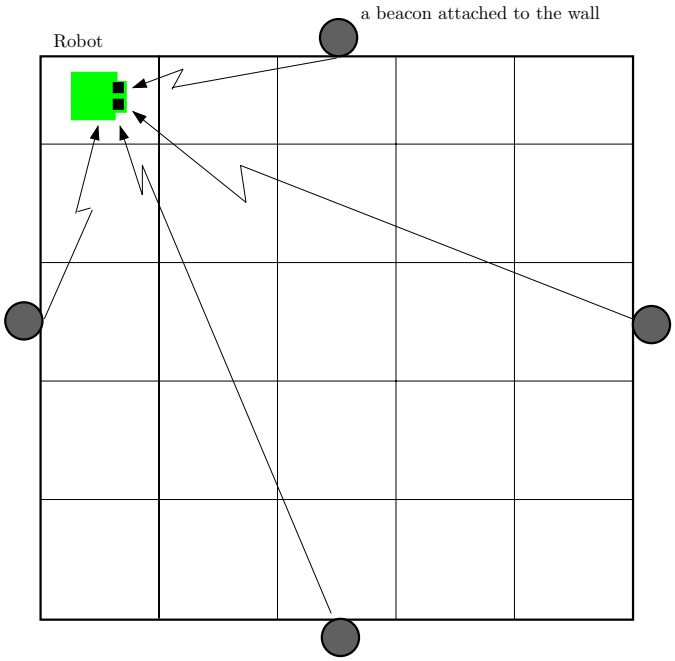

Figure 2: Robot position finding (also called localization) with the help of four transmitters (beacons) attached to each wall of the room.

Problem Statement for the Project

The objective of this project is to determine the ML estimate of the actual robot positions $\{\mathbf{s}_k\}$ of robot positions for the complete walk of the robot based on the observed sequence of positions $\{\mathbf{r}_k\}$. Brute-force computation of the ML estimate requires a computation time of the order N^{2K}. A computationally attractive alternative is the Viterbi algorithm which is able to solve the same problem in a computation time that is linear in K (rather than exponential). In this project, you will implement the Viterbi algorithm to determine the ML estimate of $\{\mathbf{s}_k\}$ from $\{\mathbf{r}_k\}$; that is, the objective is to find $\widehat{\mathbf{s}}_{\mathrm{ML}} \triangleq (\widehat{s}_0, \widehat{s}_1, \ldots, \widehat{s}_K)$.

Initial state of the robot is always known, which can be assumed to be (0,0) in this project. However, the final state may or may not be known .We shall

consider the problem for both cases.

A few test sequences $\{\mathbf{r}_k\}$ for $N = 5$ and $K = 100$ are provided at the course website in the project section. First, you should compare your results with them and then you can generate your own observation and actual state sequences for farther inquiries. Be sure to provide your solution for both the cases: when the final state is known, and when the final state is unknown.

Deliverables

1. The code for your solution that you have verified for test sequences. Write your code such that it can take N, K and the transition probabilities as input.

2. The project report that describes the approach, provides the steps to derive Viterbi algorithm, and describes your implementation. Discuss results for the observation sequences provided to you and any other sequences that you generated for yourself. Comment on the cases of known vs. unknown final state.

You are encouraged to include any alternative ideas that you tried or could incorporate at various stages of the problem solving.

Hints:

To get more information about the Viterbi algorithm, you may visit some internet sites, such as

`http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/`
`html_dev/viterbi_algorithm/s1_pg1.html`
`http://en.wikipedia.org/wiki/Viterbi_algorithm`
`http://www.cim.mcgill.ca/~latorres/Viterbi/va_main.html`

We have studied Maximum-Likelihood (ML) method in quite detail and this problem can also be approached with the same mindset. The following hints may prove helpful in giving you a starting direction:

i. Formulate the problem as ML.

ii. Take the logarithm of the likelihood function noting that log is a monotonic function.

iii. Manipulate the above expression until you end up with:

$$\hat{\mathbf{s}} = \arg \min_{\forall \mathbf{s} \in \mathcal{S}} \sum |r_k - s_k|^2$$

where $\mathcal{S}$ is the set of all the valid sequences of position that can be solution candidates.

iv. Use Viterbi algorithm to find solution to the above equation.