Robot localisation with HMM based forward-filtering

Pablo Correa Gomez - pa1850co-s - 960924-T154

March 2, 2018

Abstract

The problem that the assignment is trying to solve is how to find the placement of a robot in a room, subdivided in a squared grid, when the information that we get from the environment is not accurate. That means that we are dealing with contingency problem, where we know the state space but we get updated information (a sensor read) about our current state every time the robot executes a movement. In addition, the information is non deterministic, so can only guess the probabilities of being in a certain place.
To check the model of the robot and the placement solving algorithm, we are going to simulate the movement based on previous experience probabilities or movement model.
The algorithm used to compute the probabilities where the robot might be is the Hidden Markov Models forward-filtering. The most important remark about the Hidden Markov Models is that the probability of being in a certain state after certain amount of movements is only dependent on the previous state (abstraction from reality, as it actually depends on all previous states). This assumption is used in our problem for forward filtering, which consists in getting the probability distribution of possible states based on the previously known probability distribution and an update of the sensor. In our case, we will start with a uniform probability distribution and then simulate all the process *forward*.

Methodology

First thing to take into account is the sensor that retrieves the information from the environment. For the assignment the sensor model is defined as follows:

- Probability of the sensor giving the real position: 0.1.
- Probability of the sensor giving a position one square away from real position = 0.05 for each of the possibilities.
- Probability of the sensor giving a position two squares away from real position = 0.025 for each of the possibilities.

In consequence, the probability of the sensor failing and not giving a read is, in our case, the remaining probability That is, between 0.1 and 0.625, depending on how far from the walls the robot is.
However, this is just one interpretation of the sensor model. Other possibilities could consider the probability of failure being fixed and then correcting the other ones, or considering failure probability of 0.1 and correcting all of them so they add up to 1.

Next thing to take into account is the transition model or the robot movement model. It is defined in the assignment the following way:

- Before every movement the robot picks a direction to move based on the given probabilities:
  - Probability of continuing in same direction given that it does not encounter a wall = 0.7
  - Probability of picking a different direction given that it does not encounter a wall = 0.3
  - Probability of picking a different direction given that it encounters a wall = 1.0 (so it will not continue straight if there is a wall)
- Then the robot moves in the direction picked one position into the grid.

The conclusions extracted from this transition model are that the robot is smart enough to avoid getting into the walls and that, with a lack of information from the environment, we can conclude that the robot will most likely continue straight forward.

However, there are still some assumptions that needed to be made before the transition model could be implemented:

- If the robot does not have a wall ahead and there are more than one possibility to change direction (the robot can move right, left and go back), the possibility will be equally distributed between those possible directions

- The robot is smart enough to avoid walls even when the wall is not ahead of it. That means that if the robot is moving alongside a wall, it will never try to turn to face the wall.
- At the beginning we don't know where is the robot, so the initial probability is equally distributed between all the different positions.

To translate the transition and sensor model to code, it is needed to define some rules so that any current position is represented by a row and any future position is represented by a column. The decision taken in our solution is that the index 0 of the matrix corresponds to the top left corner in the grid with robot facing North. Subsequent indexes iterate first through directions, then through the columns of a row and, finally, through rows.

Results and Discussion

The forward filtering method gives some accurate results after a certain amount of steps, usually around 70-80. These initialization steps are needed because at the beginning we don't have any clue about the initial position of the robot, so it takes some time before the sensor reads and the transition model can give some accurate information. This could be avoided if the initial position was known.
Measuring the results in behalf of the Manhattan distance gives a value around 1.8 after those initialization steps.

Another important remark that we get from the project is that after the initialization steps, a checker pattern can be clearly observed, so that any point with some high probability is surrounded by points with probability nearly 0 and points with unnoticeable probabilities are surrounded by other with quite high ones. The main for that pattern is the movement model, that promotes the probability alternation between adjacent points of the grid. This is because once we have found a high probability of the robot being in a certain place (thanks to the sensor model), the remaining probability at that point becomes zero during the next step. This has an important consequence when dealing with the measurement of the results: **The Manhattan distance between the actual robot and the most probable position is always a multiple of 2** once the initialization has been done.

The last important remark to discuss is the behavior of probabilities when the sensor gives a failure read. In our implementation there are two main consequences:

- The number of positions where there is some probability of finding the robot is increased, as the robot might have moved further.

- The probability of finding the robot close to walls and corners is increased. This is due to our sensor model, where a failure is a lot more probable closer to walls than in the middle of the grid. This make updates completely necessary even when the sensor fails, while in a regular implementation of the forward filtering problem, it can be avoided, as there is no information supplied.

Implementation

The code and compiled program can be found in the machine login.student.lth.se in the following folder: **/h/d8/y/pa1850co-s/workspace/hmm/src**. It can be executed by invoking **java control.Main**, and it will generate an 8x8 grid as requested in the assignment.

As a matter of academic honesty, the implementation to make a random decision based on the probability distribution has been taken from Stack Overflow proposal[1].

[1] `https://stackoverflow.com/questions/9330394/how-to-pick-an-item-by-its-probability`