Robot localisation with HMM based forward-filtering

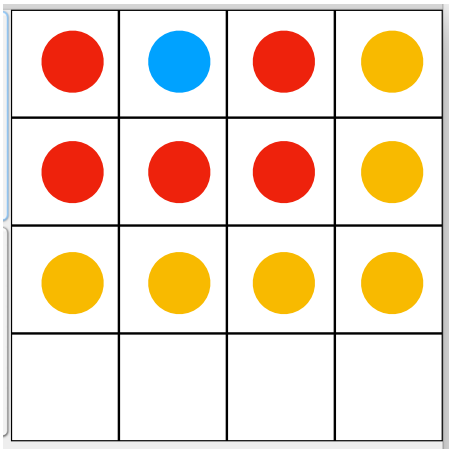
This task is essentially corresponding to task 15.9 in the course book, with a more detailed specification of the sensor model. The task relies on the explanations for matrix based forward filtering operations according to section 15.3.1 of the book.
Hence:
If you do not have access to this particular edition (Artificial Intelligence: A Modern Approach, 3/e, by Stuart Russell and Peter Norvig, ISBN-10: 0132071487) and thus have difficulties in figuring out what to do, please contact me (Elin) at Elin_Anna.Topp@cs.lth.se before attempting to solve the task!

In short

You are assumed to work with robot localisation based on forward filtering with a Hidden Markov Model.

In detail

You are supposed to implement an HMM to do forward filtering for localisation in an environment without any landmarks. Consider the (in the book) previously mentioned vacuum cleaner robot *in an empty room*, represented by an *n x m* rectangular grid. The robot's location is hidden; the only evidence available to you (the observer) is a noisy sensor that gives a direct, but vague, approximation to the robot's location. The sensor gives approximations $S = (x',y')$ for the (true) location $L = (x, y)$, the directly surrounding fields *Ls* or the "second surrounding ring" *Ls2* according to the specifications below. Here, *n_Ls* is the number of directly surrounding fields for *L* (this can be 3, 5, or 8, depending on whether *L* is in a corner, along a "wall" or at least 2 fields away from any "wall") and *n_Ls2* is the number of secondary surrounding fields for *L* (this can be 5, 6, 7, 9, 11, or 16, depending on where *L* is located relative to "walls" and corners).



In the example in the figure, the blue spot marks *L*, the red spots are the in this situation possible 5 positions in *Ls,* and the yellow dots mark the 6 possible positions in *Ls2.*

The sensor reports

- the true location *L* (blue) with probability 0.1
- any of the *n_Ls* ∈ {3, 5, 8} existing surrounding fields *Ls* (red, here *n_Ls* = 5) with probability 0.05 each
- any of the *n_Ls2* ∈ {5, 6, 7, 9, 11, 16} existing "secondary" surrounding fields *Ls2* (yellow, here *n_Ls2* = 6) with probability 0.025 each
- "nothing" with probability 1.0 - 0.1 - *n_Ls**0.05 - *n_Ls2**0.025.

This means that the sensor is more likely to produce "nothing" when the robot's true location is less than two steps from a wall or in a corner (there are also other possibilities of setting up the sensor model, but you should stick to this model for the implementation).

The robot moves according to the following strategy:

Pick random start heading h_0. For any new step pick new heading h_t+1 based on the current heading h_t according to:

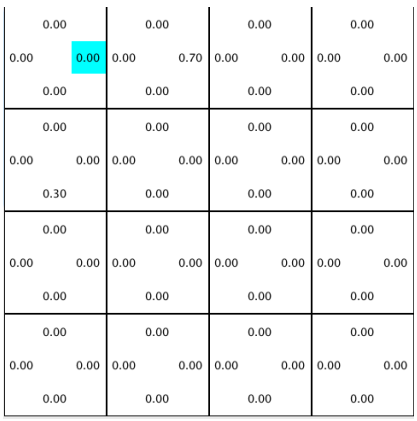
P(h_t+1 = h_t | not encountering a wall) = 0.7
P(h_t+1 != h_t | not encountering a wall) = 0.3
P(h_t+1 = h_t | encountering a wall) = 0.0
P(h_t+1 != h_t | encountering a wall) = 1.0

It then moves in the direction h_t+1 by one step in the grid. This means essentially that it a) will always move one step and b) it can only move straight.

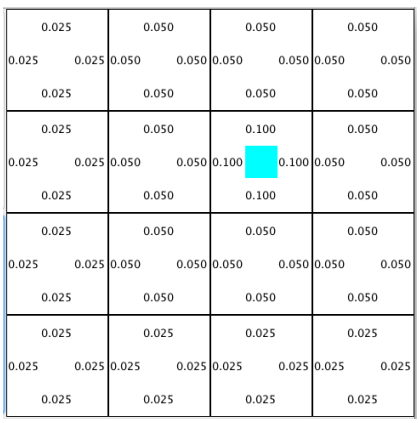
In case a new heading is to be found, the new one is randomly chosen from the possible ones (facing a wall somewhere along the wall leaves three, facing the wall in a corner leaves two options of where to turn, see one example below in hint 1).

Implement this as an HMM (according to the matrix-vector notation suggested in section 15.3.1 of the course book) and apply **forward filtering** to track the robot. This requires obviously a two-part implementation, as you need to simulate the robot and its movement (from which you also can simulate the sensor readings) to have some ground truth to evaluate your tracking against, and the HMM-based tracking algorithm as such. Your algorithm should basically loop over the following three steps:
1) move (simulated) robot to new pose according to movement model
2) obtain (simulated) sensor reading based on its actual position given the sensor model
3) update the position estimate (vector f) based on the sensor reading from step 2, using the known sensor and transition models

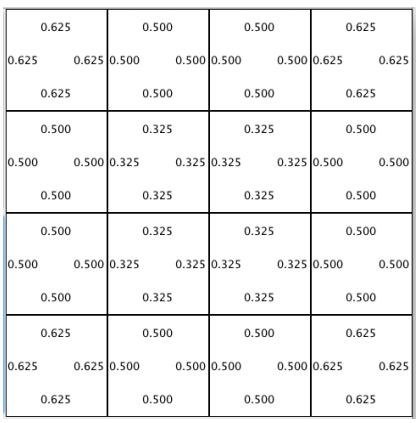
Hint 1: Encode the states such that each possible state represents the position of the robot in the grid plus a heading. This means you will have rows * columns * 4 possible states, and rows*columns+1 possible sensor readings. Your transition matrix will thus have the dimensions (rows*columns*4) x (rows*columns*4) and you will need (rows*columns+1) observation matrices, each being a diagonal matrix of dimensions (rows*columns*4) x (rows*columns*4) to get the matrix multiplications to work out. However, note the following: It might make sense to *store* the observation matrices in vectors, representing the diagonal of the respective matrix. Also, the filtering algorithm only knows sensor readings, which only represent positions in the grid, i.e. one sensor reading is equally likely for the four states (headings) that correspond to one position in the grid, thus, the values within each "bundle" of four consecutive entries along the observation matrix diagonal should be the same (another option for saving space).



TRANSITION VISUALISATION = Übergangsmatrix



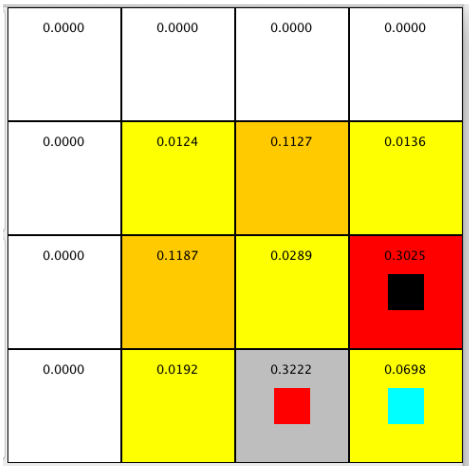
OBSERVATION MATRIX



In the three figures above you see a grid visualisation (each cell (x, y) has four headings, representing a bundle of four states with DIFFERENT probabilities to be reached from a specific state, but with the SAME probability to have caused a certain sensor reading) of one row of the transition matrix and the diagonal of one observation matrix. The transition visualisation (left) shows the probabilities to go from (0, 0, EAST) (cyan) to any other state, i.e. only two states, (0, 1, EAST) and (1, 0, SOUTH) are possible and have a probability larger than 0. In the observation matrix examples (middle and right), you see the probabilities for each respective state (x, y, h) to have generated the (cyan) reading r = (1, 2) (middle), and the probabilities for each respective state (x, y, h) to have generated "nothing" (right).

Hint 2: Even though a sensor reading of "nothing" normally means to do the forward step without update, i.e. it boils down to mere prediction, you should use the information given in the known sensor model as stated above, i.e. a sensor reading of "nothing" is slightly more likely to get when the robot is close to a wall. Thus, even a "nothing" reading from the sensor should entail a proper prediction + update step.

Hint 3: Assume a grid size of preferably 8x8 (at least, however, 5x5) to base your evaluation on. If you use Java and an 8x8 grid, you should observe something like 30-35% of correct estimates rather quickly, roughly 100 steps should already get you there safely. The average Manhattan distance (i.e. the number of "robot-steps" necessary to get from estimated to true position) should then be somewhere between 1.6 and 2.0. It is quite common to observe a checker-board pattern in the visualisation (see to the right, where the darker colours stand for higher, lighter for lower probabilities, grey is the highest one found, black is the true location and cyan marks the current sensor reading).



You can find a zip-file with a Java-based tool for visualisation of **your** transition model, **your** sensor model and your estimation (including some sort of user's guide) included in this archive! Feel free to use other programming languages (C++, Matlab, Python are ok as well) or tools, but if you choose Java, please make use of the visualisation - it presumably helps me a lot to understand your implementation!

If you experience difficulties with understanding the task, setting up the models, getting the implementation to work, etc, please, CONTACT us well BEFORE the deadline (see below)!

Write a brief report on your approach, explaining your thoughts on the model and implementation, and, even more important, **discuss** the results.
Use the following questions as a skeleton for your reflections and discussion!

Q1: How (other than requested for this task) could one interpret the original definition (in the book) of the sensor model, i.e. correct with p=0.1, "one step off" with p=0.05, "two steps off" with p=0.025, and reporting "nothing" with p=0.1?

Hint 4: Look at what happens when the robot is close to “walls” and “corners”.

Q2: How accurately can you track the robot’s path with this approach?

Hint 5: In terms of robot localisation it is often not relevant to know how often you are 100% correct with your estimate, but rather, how far "off" your estimate is from reality on average / how often. You could measure the distance between true location and estimate by using the Manhattan distance (how many robot steps off), or the direct Euclidean distance (looks nicer, but would not help a robot that can only move straight too much).

Report guideline: The report should roughly follow the outline of a short scientific article (2-3 pages), i.e. it should have an **introduction** explaining the problem to solve (self-contained and in your own words) and the general approach to be used, a section on your **method**, i.e. an explanation of the sensor model that also considers an answer to the above first question, how you implemented the transitions, assumptions you made, etc., a **result** section and finally a **discussion**, in which the second question should be answered. Additionally there should be one section on the **implementation**, where to find and how to run it. The report should then be submitted as a **PDF**-document (even Word can produce those!).

(**IMPORTANT NOTES:** Please have your final, runnable version available on the student machines (LINUX!) in E-house, we do not want to have to download / clone, organise and compile all your roughly 60-70 submissions. The path quoted to describe where to find your code must include your username and you have to make sure it is accessible for the person evaluating your submission. Check the Linux chmod command for that purpose, please. The code should be accessible both for reading and running without any additional demands, like compilation or usage of some particular IDE.)

You may consult the code in the textbook code repository (or any other implementations), but the code you hand in must be primarily your work. You should not provide any code printout in the report.

The **deadline** for this assignment is Friday, 2nd of March, by which you MUST file in your working solution to the assignment (i.e. testing of your implementation according to your guide of usage should be possible).

Important remark: Remember that the time frame allotted to this assignment is approximately three days of work, so please don't overdo it: a correctly working program together with a complete report according to above guideline will suffice to get a pass.

If you decide to use someone else's code (e.g. some library found on the web), please mention it both in the code and in your report: it is a matter of academic honesty. Lund University is committed to fighting every case of dishonesty or plagiarism.

The report should be sent to tai@cs.lth.se as an attachment to a mail message with the following subject line:
Assignment 3 by *username1* and *username2*,
where *username{1,2}* are your user names in the student computer system, like dat14eto.

If you fail to adhere to this, you might have to wait arbitrarily long for your submission to be noticed and commented!

The resulting programs should remain in your directory until you have been notified of the result, e.g. on the notice board and/or web or by e-mail. You may expect that your report and implementation will be examined within two weeks. If your report or implementation is unsatisfactory you will be given one chance to make the corrections and then to hand it in within a week after you have been notified (on the notice board and/or web or by e-mail).

If you need **help** or **advice**:
The course assistant Erik Gärtner (erik.gartner@math.lth.se) is available in his office in room Mh: 352 in the MATH-building on Tuesday, February 20 from 2pm to 5pm and on Wednesday, February 28, from 2pm to 5pm, otherwise you can try to get your questions answered by e-mail.

If you want to meet with Elin Topp (the responsible teacher) personally, try her office E:4128 in E-house (preferably on Tuesdays, 9:30 - 11:30), if she is not there or you cannot make it during that time slot, make an appointment by e-mail (elin_anna.topp@cs.lth.se).