Artificial Intelligence, Assignment 3

Probabilistic Reasoning Over Time

Name: Isabelle Frodé

Date: March 3 2021

1 Implementation

1.1 Summary

The task of this assignment was to localise a robot in an environment without landmarks. This was solved by implementing a Hidden Markov Model that uses forward filtering for localisation, as explained in *Artificial Intelligence A Modern Approach* by Stuart Russel and Peter Norvig [1].

The robot could be considered to be placed in an empty room simply represented as a *n x m* grid. The location of the robot is non-observable. There is however observable evidence of the robot's location. These observations are somewhat noisy approximations, generated by a sensor.

The first part of the implementation aimed to simulate the robot's movements. The second part consisted of implementing the localisation algorithm using the Hidden Markov Model. Finally, an evaluation of the model was implemented. The algorithm loop over these 4 steps:

1. Simulate movement of robot using movement model
2. Simulate sensor readings using according to the sensor model
3. Update position estimate using forward-filtering
4. Evaluate

1.2 Peer review and Implementation Discussions

The peer review was done with Moritz Tanneberger. During the implementation I discussed with Felicia Segui and Louise Karsten. The peer review let me to rethink my implementation. I detected and fixed some bugs since the code was not running correctly, I also made sure that the robot could not move outside the grid in the move phase. After the review the "no read" by the sensor was also added.

1.3 Explanation of Models

1.3.1 Transition Model

The transition model describes the probability of the agent to move from one state to another. The model is fixed because it describes a stationary process. The transition matrix T is an SxS matrix where S is the number of possible states an agent can be in. The transition matrix $\mathbf{T_{ij}}$, stated in equation 1, describes the probability of the robot to move from state i to state j. X_t is the state variable representing a location on the grid, NEIGHBOURS(i) set of squares adjacent to i and N(S) the size of the set [1]

$$\mathbf{T_{ij}} = \mathrm{P(X_t|X_{t-1})} = \begin{bmatrix} \mathrm{T_{11}} & \mathrm{T_{12}} & \dots & \mathrm{T_{1j}} \\ \mathrm{T_{21}} & \ddots & & \\ \vdots & & \ddots & \\ \mathrm{T_{i1}} & & & \mathrm{T_{ij}} \end{bmatrix} = \frac{1}{N(S)} \text{ if j} \in \text{NEIGHBORS(i) else 0} \tag{1}$$

The visualisation of the transition model show that the probabilities of the robot going to adjacent squares are uniform, meaning that the agent is equally likely to end up in any neighbouring square.

1.3.2 Sensor Model

The sensor model describes the probabilities of a state to cause an evidence e_t to appear at time t, where n is the number of states an agent can be in. The sensor matrix $\mathbf{O_t}$ contains the sensor readings for all states [1]:

$$\mathbf{O_t} = \begin{bmatrix} \mathrm{P(e_t|X_t = 1)} & 0 & \dots & 0 \\ 0 & \mathrm{P(e_t|X_t = 2)} & & \\ \vdots & & \ddots & \\ 0 & & & \mathrm{P(e_t|X_t = n)} \end{bmatrix} \tag{2}$$

Since this process is stationary the sensor readings are fixed. The sensor model returns 1) the probabilities of the sensor to report the given position, and 2) the probability of each state to have generated "nothing" in terms of reading. The visualisation show that "No read" is more likely to occur for states close to the wall or a corner, and that the sensor is more likely to report the true state and states adjacent to the true state than further away.

1.3.3 Filtering process

The Forward Filter is used to track and localise the agent by calculating the probability distribution for each time step. The forward-filtering algorithm is given by equation 3, where α is a normalization factor, $\mathbf{O}_{t+1}$ is the sensor matrix, $\mathbf{T}^{\mathrm{T}}$ the transition matrix in transpose form and $\mathbf{f}_{1:t}$ the previous distribution model [1].

$$\mathbf{f}_{1:t+1} = \alpha\, \mathbf{O}_{t+1} \mathbf{T}^{\mathrm{T}} \mathbf{f}_{1:t} \tag{3}$$

The visualisation show that every time the agent moves one step the filtering process updates, showing a new probability distribution for each time step.

1.4 Results and Discussion

The results show that the hit rate of the implementation goes to approximately 0.05-0.10, after a number of steps > 100 for a grid of 8 x 8 squares.

How far off the estimate of the robot state is from the true state is given by the average Manhattan distance. After a large number of steps the average Manhattan distance goes to approximately 2 robot steps.

Pure guessing as estimation of the state would have caused an expected value of the hit rate of approximately 0.015 for an 8 x 8 grid for a large number of steps. The algorithm produces approximately 4 times better estimates than pure guessing. The Manhattan distance to the true state could be assumed to be approximately 5.25 for pure guessing. By implementing only the sensor readings in the implementation and not the ”no read”, assuming a perfect sensor, the accuracy increases to a hit rate of approximately 0.10-0.15. The Manhattan distance would not be affected.

2 Article Summary

The article [2] describes Monte Carlo Localization (MCL), which is an algorithm within the Markov localization family. The motivation to the MCL algorithm is that there are problems with the previously used Markov localization techniques. For example it takes up a lot of space and memory and it require high performance computing. The article presents empirical results showing that the MCL algorithm outperforms the old algorithms in several ways. For example it focuses high cost computing to regions with high likelihood, hence it is more accurate and requires less computation. It is also possible to track the robot globally instead of just locally, and it is easier to implement than previous methods.

3 Discussion

The results from the implementation of the HMM algorithm show that the estimates of the robot's states are much better with the HMM algorithm than without (just guessing). I would however argue that the MCL algorithm actually should be considered instead. My implementation had a hit rate of approximately 5-10 %, which one would argue is not accurate enough in many cases. I am nevertheless not certain that my results are correct nor perfect after having some difficulties with the implementation at times. This lead me to the other argument, the MCL algorithm is actually ”much easier to implement” than the previous Markov model algorithms [2]. The MCL algorithm could help to more accurately track the robot and it scales well. Additionally it is easier to implement and therefor I would consider implementing the MCL algorithm instead.

References

[1] P Norvig, S Russel. *Artificial Intelligence: A Modern Approach. 3rd ed.* 2010. Upper Saddle River, NJ: Prentice Hall

[2] D Fox, W Burgard, F Dellaert, S Thurun. *Monte Carlo Localization: Efficient Position Estimation for Mobile Robots.* 1999. Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99)