6.834J/16.412J, S15
Professor Brian Williams

Problem Set #4

Due: May 13, 2015

Dustin Tran, Xiaomin Wang, Rodrigo Gomes
{trandv,xiaominw, rgomes}@mit.edu

Task: Robot grocery shopping in partially observable settings

1 Motivation

Imagine that you’re in your bed, hungry, and you just don’t want to walk all the way to the grocery store. Equivalently, imagine you’re working, doing the most exciting task to save humanity, and you have no time to grab food—menial tasks are beyond you. We’d like a robot which can intelligently learn to shop groceries for us: it understands a query from the user, moves to the grocery store, finds the relevant items, purchases it, and comes back to the user.

In our project we focus on the arguably most difficult of these tasks, which is to locate the groceries in the store. Moreover, we work under the realistic scenario in which the robot can only observe its surroundings (POMDP) rather than an understanding of where it precisely is in the store (MDP).

Given this partially observable setting, the robot should learn how to obtain all items in the grocery store and do so in an optimal amount of time. It must 1. figure out where it is in the supermarket; 2. intelligently search for the items by learning which aisle corresponds to which category; and 3. find the optimal path and sequence to obtain all items.

2 Setup

The robot is given a list of items it should find in the supermarket. The supermarket is represented by a grid world. It has several aisles containing different categories of goods. The robot has a map of the supermarket (it knows where the walls and aisles are), but it does not where items are located on the shelves. The robot has a perfect sensor, which can observe its four neighbors. It can move in four directions, up, down, left, and right. It will transition to the intended position 90% of the time. For the rest of the time, it is equally likely to move to any of the permitted direction, except the one opposite the intended direction. The robot is equally likely to start the mission in one of the four corners of the supermarket. When the robot goes to the grid next to a target item, the item is considered found. It continues moving untill all the items have been found.

3 Procedure

The code for the grounded scenario can be found in the `visual` folder. You can start the simulation by running `python debug.py`. An instance of the robot, and random supermarket is created automatically. To set what products you want the robot to grab for you, you can write write: `g.targets = set([...])`. For example, if you want ice cream, and beans, you can write: `g.targets = set([’iscream’, ’beans’])`. To view all possible products, organized by aisles, just type: `print g.aisles_content`. After selecting your items, just type `go()` to start the robot demo.

3.1 Belief Update

At every step of the demo, the following happens:

- Agent provides next action based on current belief state
- Simulator executes action (errors happen with some probability)

- Belief state is updated based on transition probabilities
- Belief state is updated based on observation
- Belief about the world is updated based on belief state, and observation

The belief updates are done using Bayes Rule. One issue we ran into is that we are making the Markov assumption for our states, which is not entirely correct, due to the fact that the organization of the aisles is part of it. This causes us to be biased towards our most likely state when entering an aisle, since each step causes that aisle's type to be more likely, and the most likely state to, in turn, become more likely. The desired behavior would be to only update the aisle organization belief state when we enter the aisle, and not as we walk in it. An alternative implementation of our belief state that could potentially avoid this problem would have been a particle filter.

3.2 Max Probability Value Iteration

We reduce the POMDP to MDP, and solve the MDP using value iteration. At each time step, we assume that the robot is at the most likely state based on its belief state. We also generate an arrangement of the all items according to the probability distribution of items on the aisles. With this information, the robot knows exactly where it is and where everything is. Since it also has the transition model, value iteration becomes straightforward. Thus we can find the best action to take assuming those are the true state of the world.

In a way, we are approximating the POMDP with most likely state MDP. Alternatively, we can also sample over all possible state according to their probability distribution and run value iteration for each of them. We then choose the best action that gives the robot the maximal weighted averaged expected reward. Such sampling can lower the probability of choosing a bad action assuming a completely wrong world state. Thus the robot can make smarter moves, but at the expense of computation power.

4 Experiments

4.1 Path Planning

The robot plans its path intelligently but greedily. As long as there are still targets to be retrieved, the robot moves to the state with highest reward. But it does not try to minimize the total distance of retrieving all the targets. If the best state at the moment is very far away, but there is a target with lower reward closer to the robot, but not on the way to the best state, the robot will ignore the closer target and go for the best state.

One approach to address this non-optimality is to include the remaining target items as part of our state. The reward function will also have to be changed accordingly. However, this will increase the state space greatly and slow down the value iteration.

4.2 Run Time

Value iteration runs fairly fast for our problem size. The number of iterations it takes to converge depends a lot on the value of the discount factor, γ. For example, for a 11 by 7 grid and a γ value of 0.9, it takes about 90 iterations. If we change γ to 0.5, it only takes about 15 iterations.