Car Tracking report

071125 葉長瀚

Part I. Implementation:

Part 1:

```
53  def observe(self, agentX: int, agentY: int, observedDist: float) -> None:
54      # BEGIN_YOUR_CODE (our solution is 9 lines of code, but don't worry if you deviate from this)
55      for i in range(self.belief.numRows):
56          for j in range(self.belief.numCols):
57              y=util.rowToY(i)
58              x=util.colToX(j)
59              dis= math.sqrt((agentX-x)**2 + (agentY-y)**2)
60              p=self.belief.getProb(i,j)*util.pdf(dis, Const.SONAR_STD, observedDist)
61              self.belief.setProb(i,j,p)
62      self.belief.normalize()
63      # END_YOUR_CODE
```

$$\mathbb{P}(H_t|E_1 = e_1, \cdots, E_{t-1} = e_{t-1})$$

to $\mathbb{P}(H_t|E_1 = e_1, \cdots, E_t = e_t) \propto \mathbb{P}(H_t|E_1 = e_1, \cdots, E_{t-1} = e_{t-1})p(e_t|h_t)$

Update the probability on each tile by multiply the original one with the emission probability got from *util.pdf*, and normalize the updated probability.

Part 2:

```
85      def elapseTime(self) -> None:
86          if self.skipElapse: ### ONLY FOR THE GRADER TO USE IN Part 1
87              return
88          # BEGIN_YOUR_CODE (our solution is 10 lines of code, but don't worry if you deviate from this)
89          def z():
90              return 0
91          p_dic=collections.defaultdict(z)
92          for (old,new) in self.transProb:
93              p_dic[new]+=self.belief.getProb(old[0],old[1])*self.transProb[(old,new)]
94          for i in range(self.belief.numRows):
95              for j in range(self.belief.numCols):
96                  self.belief.setProb(i,j,p_dic[(i,j)])
97          self.belief.normalize()
98          # END_YOUR_CODE
```

In this part, you will implement a function elapseTime that updates the posterior probability about the location of the car at a current time step t

$$\mathbb{P}(H_t = h_t|E_1 = e_1, \cdots, E_t = e_t)$$

to the next time step $t + 1$ conditioned on the same evidence:

$$\mathbb{P}(H_{t+1} = h_{t+1}|E_1 = e_1, \cdots, E_t = e_t)$$
$$\propto \sum_{h_t} \mathbb{P}(H_t = h_t|E_1 = e_1, \cdots, E_t = e_t)p(h_{t+1}|h_t)$$

Create a dictionary with default value 0 to store the new probability updated by transition probability. Sum the new probability in each tile, set them to be the belief probability and normalize it.

Part 3:

```python
def observe(self, agentX: int, agentY: int, observedDist: float) -> None:
    # BEGIN_YOUR_CODE (our solution is 12 lines of code, but don't worry if you deviate from this)
    for po in self.particles:
        dis= ( (util.rowToY(po[0])-agentY)**2+ (util.rowToY(po[1])-agentX)**2)**0.5
        self.particles[po]*=util.pdf(dis, Const.SONAR_STD, observedDist)
    p = collections.defaultdict(int)
    for _ in range(self.NUM_PARTICLES):
        p[util.weightedRandomChoice(self.particles)] += 1
    self.particles=p
    # END_YOUR_CODE
    self.updateBelief()
```

Update the probability by multiply it with the emission probability for all the particles. Create a new dictionary to store new particles, resample a same amount of particles according to the reweighted probability, and set the particle distribution to be the new one.

```python
def elapseTime(self) -> None:
    # BEGIN_YOUR_CODE (our solution is 6 lines of code, but don't worry if you deviate from this)
    new_p=collections.defaultdict(int)
    for p in self.particles:
        for _ in range(self.particles[p]):
            new_p[util.weightedRandomChoice(self.transProbDict[p])]+=1
    self.particles=new_p
    # END_YOUR_CODE
```

Update particles by the transition probability. For each particle in each tile, resample its next most probable location base on the transition probability dictionary, and then set the particles distribution to be the new one.

Problem faced and solved:

The most difficult problem I faced during this homework is indeed as the spec mentioned, that is to understand the concept of Bayesian network in order to correctly update the probability, so before start coding, I watch some video and review the course slide to ensure my comprehension.