Eda Gür

50488

Comp 341 – HW4 Report

Q1-) Pacman is not observing the ghost. So, the ghost's actions will not have an effect on Pacman's beliefs. Instead, Pacman's beliefs are based on the geometry of the board and the actions of the ghost. That is why Pacman's probabilities settle even though the ghost keeps moving. The first ghost moves randomly. So Pacman's beliefs are more or less uniform throughout the board. However, the second ghost is a ghost that moves South. This will cause Pacman to focus his beliefs on the South of the board. This can be seen by the light blue squares on the bottom of the board in the second test case.

Q2-) The third test case can find the ghost, because it can move and thus change his point of view. Howeveri the second test case can't because it can't move around its box. According to my code:

```
trueDistance = util.manhattanDistance(pacmanPosition, p)
allPossible[p] = self.beliefs[p] * emissionModel[trueDistance]
```

For a given legal position p, the trueDistance stays constant, because Pacman can't move. Consequently, the emissionModel gives the same result and we can't change our beliefs enough to decide between the four possible locations fort the second test case.

Q3-) The probabilities get reinitialized twice in test case 6 and several times in test case 7. We can see the location colors changing suddenly. This happens because all the particles have zero weights and therefore resampled from prior so that they can recover. If we had more particles, it might help by taking more time for all the particles to reach zero weight.

Q4-) With particleObserve cases, we get the probabilities faster. This is because approximate inference favors less computational time over accuracy. (There is a trade-off.) However, with exactObserved, we are guaranteed to get better/equally good results since it is exact inference.

Q5-) For each ghost, I create a new gamestate with the setGostPosition method supplied with the particles. Then, I get a ghost position distribution for that ghost. With this distribution, I can get a new sample for that ghost. After doing this for each ghost, I end up with a particle list that I can use for the next time. For the weight, I sum up emissionModels[ghost][trueDistance] over all ghosts for each particle. I don't think increasing the number of particles would help. I tried but fourth test case in question 6 still gets resampled 11 times.