Comp341
Homework #4
Can Gözpınar 68965

Report

Q1) It settles even though, the ghost is moving because, pacman is not observing the ghost. Because of this, ghosts actions has no impact on the pacman's belief. So as time passes, according to the the prior belief of pacman, on ghost's behavior, probabilities start to converge($P_{infinity}$). This leads pacman's belief to reflect places on the board where, to his prior knowledge ghosts are most likely to be given the design of the board and the valid ghost movements that are to the knowledge of the pacman. I can tell the two ghosts apart by looking at their expected ways to move to find out about the $P_{infinity}$ to figure out which ghost is which. The probability that it converges would reflect the ghost's location.

Q2) In the case, in which we can not find the ghost, observations that we get become not enough to make us discern between the states up to that point. What I mean is that after a certain point our beliefs accumulate at certain points and the observations we get does no good on us on deciding which one of them is more likely or less likely to be the ghost's location since, probability after taking the observation into account doesn't turn the tables.

Q3) I can tell when the particles get re-initialized. When the particles get re-initialized, I observe that probability distribution becomes uniform. In other words, all of the board locations suddenly change color to become the same shade of blue. Pacman gets in this situation whenever there is an inconsistency. For example, when due to an observation all of the probabilities in the probability distribution has zero weights which renders it meaningless, it gets re-initialized. Also, when probability distribution converges to a certain location and upon getting to that location pacman sees that it has not eaten a ghost therefore it re-initializes again. Increasing the particles would not guarantee a certain solution but it might increase the accuracy of the probability distribution whereas, there is also computational cost that comes with it too. So increasing would not solve it for sure, one should aim for the most efficient particle number which is not so easy to find out given ones criterion.

Q4) Approximate inference probabilities theoretically converge to the exact inference. In our examples they are pretty close with minor differences which could stem from many factors. Approximate inference could help us out with cases in which using exact inference is not feasible due to very large CPT's or computational expensiveness. I believe approximate inference performed pretty well with 5000 particles. I think using 5000 particles makes sense since, it is pretty large compared to the size of the game boards, this increases our accuracy. Increasing the particle number might increase the accuracy in some cases but it would also increase the computational complexity so it comes with a cost. I believe a better size could be found but I cannot put it into words by simply saying larger is better nor smaller is better there is no generic number that I can to come up with those that could be applied to every problem. Overall, I believe it performed presumably well.

Q5) To deal with the new particles as time elapsed I did the following. I iterated over each particle which contained positions for the each ghost. And within each loop, I also looped through each ghost. For each ghost using the getPositionDistribution function that was supplied to me I attained the distributions for new positions for a given ghost. Then I sampled from that new distribution that I had just attained and appended to a new list that I use to hold new values for the sample on each stage until I append it then It gets re-initialized and goes through these same steps again until, all of the samples are re-sampled. Note that since I am using for loop on each sample, I append the resampled values to the new sample list which conforms to the order of the originial samples ghost ordering. After looping through each ghost within a given sample loop cycle, I get

my newParticle variable instance which is the re-sampled version of the oldParticle. Finally, after running through each particle in my self.particles I have all of the re-sampled variables stored in newParticles and all that that remains is to set the self.particles equal to the newParticles to have the particles updated with the newly re-sampled particles that we have just computed. One thing that helped a lot was that getPositionDistribution() returned DiscreteDistribution instance therefore, sample() method was available to us which made sampling from the new distribution of new ghost positions much easier to implement.