Group 2: Alex Medellin and Nick Rummel

CSC463

Dr. Girard

November 6, 2017

**Tabular Sarsa Lab**

1. There were no significant changes made to the design of our robot since the last assignment. The only change was mounting three touch sensors. One on the left and right sides of the front of our robot and one placed in the middle of the back.
2. One disadvantage is the fact that our robot can’t detect a wall to either side of it, and could be remedied by placing a sensor on each side. These additional sensors were not needed to complete the task, but could be useful when the robot is very close to a wall that it is parallel with or approaching a wall at an incredibly small angle. Given our current placement of sensors, our robot has a greater chance of colliding with a wall from the side and not detecting it until one of the front sensors registers a hit.
3. If the robot was capable of using a long-term memory, it could remember the rewards for each action between runs. In theory, the robot would learn which actions to use from each run and develop larger reward values for actions. This would eventually allow the robot to continually navigate and exit the maze more quickly than the previous runs until it has figured out the ideal path to navigate the maze. It would also potentially save time between the first and second run because only the first run would need random actions initially assigned.

|  |  |
| --- | --- |
| **Action** | **The robot will:** |
| *TurnLeftAt45* | turn 45 degrees to the left. |
| *TurnLeftAt90* | turn 90 degrees to the left. |
| *TurnRightAt45* | turn 45 degrees to the right. |
| *TurnRightAt90* | turn 90 degrees to the right. |
| *Turn180* | face the opposite direction (180-degree turn). |
| *Forward* | move forward. |
| *Backward* | move backwards. |

|  |  |
| --- | --- |
| **Reward** | **Reward Value** |
| *isCollidingWithBothFrontSensorsAfterAction* | - 0.5 |
| *isNotCollidingWithBothFrontSensorsAfterAction* | + 0.5 |
| *isCollidingWithLeftSensorAfterAction* | - 0.3 |
| *isNotCollidingWithLeftSensorAfterAction* | + 0.3 |
| *isCollidingWithRightSensorAfterAction* | - 0.3 |
| *isNotCollidingWithRightSensorAfterAction* | + 0.3 |
| *isCollidingWithBackSensorAfterAction* | - 0.2 |
| *isNotCollidingWithBackSensorAfterAction* | + 0.2 |
| *isCollidingWithBothFrontSensorsAfter4Sec* | - 0.3 |
| *isNotCollidingWithBothFrontSensorsAfter4Sec* | + 0.3 |
| *isCollidingWithLeftSensorAfter4Sec* | - 0.2 |
| *isNotCollidingWithLeftSensorAfter4Sec* | + 0.2 |
| *isCollidingWithRightSensorAfter4Sec* | - 0.2 |
| *isNotCollidingWithRightSensorAfter4Sec* | + 0.2 |
| *isCollidingWithBackSensorAfter4Sec* | - 0.1 |
| *isNotCollidingWithBackSensorAfter4Sec* | + 0.1 |

1. Given some time, our robot would most likely make it out of the maze. In our demonstration, the robot did learn enough to exit the maze. The code we created keeps a count of how many times each action has been used. If the robot seems to be favoring that action repeatedly, we try to keep the robot from using that action. This allows the robot to try a new action and learn about the next part in the maze loop.