Practical 4: Reinforcement Learning

1 Introduction

In this practical, we try to learn the optimal policy for the monkey in the game Swingy Monkey. Since we don't know the probabilistic transition model or the reward model, we use a reinforcement learning approach.

2 State Space

We tried several things to determine the optimal state space for reinforcement learning. First, we note that the set of possible positions in the space is very large, such that it was necessary to discrete the space into larger blocks rather than using individual spaces. Second, the state at any given time includes the distance to the next tree trunk, the screen height of the top of the tree trunk gap, the screen height of the bottom of the tree trunk gap, the velocity of the monkey, the screen eight of the top of the monkey, and the screen height of the bottom of the monkey.

The first configuration we tried was a tuple of the difference between the height of the top of the monkey and the height of the top of the tree trunk gap and the distance to the nearest tree. However, we were unable to achieve a score of more than 10-15 after 100 iterations. We observed that most of the deaths were being caused, not by running into a tree, but by falling off the screen, and thus altered our state configuration to take this into account. Our final state space representation is a tuple of four elements: a binary of whether or not the monkey is near the top of the screen (a height of the top of the monkey greater than 250 pixels), a binary of whether or not the monkey is near the bottom of the screen (a height of the bottom of the monkey less than 50 pixels), a binary of the velocity (whether the velocity is greater than 0), and the difference between the top of the monkey and the top of the nearest tree binned into 20x20 blocks.

3 Q-Learning

Q-Learning is a model-free reinforcement learning algorithm that uses the expected value of taking action a under state s under the optimal policy from the next state onwards to determine the next action. From the Bellman equations, we know that the Q value function is as follows:

$$\begin{aligned} Q(s,a) &= R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a' \in A} Q(s',a') \\ &= R(s,a) + \gamma \mathbb{E}_{s'}[\max_{a' \in A} Q(s',a')] \\ &= \mathbb{E}_{s'}[R(s,a) + \gamma \max_{a' \in A} Q(s',a')] \end{aligned}$$

Thus, we estimate $Q(s,a)$ as the expectation over the state space. Since we don't have the probabilistic transition function, we update our $Q(s,a)$ value every time we take an action a from state s and receive a reward r. We thus get a sample from the distribution $P(s'|s,a)$. We thus update using temporal difference learning, in which the new estimate is adjusted to reduce the difference to the target value in the state reached in the next iteration. Thus our update takes the form:

$$Q(s,a) = Q(s,a) + \alpha[(r + \gamma \max_{a' \in A} Q(s',a')) - Q(s,a)]$$

where $0 < \gamma < 1$ is the discount factor and $0 < \alpha < 1$ is the learning rate.

We note that Q-Learning has two properties that dictate how it converges to the optimal policy. First, if we exploit the Q-values in the limit, that is, always choose the action that has the greatest expected value, the policy will converge to the optimal policy. Second, if we use a learning rate specific to each state-action

pair, or $\alpha_k(s,a) = 1/k$ where k is the number of times action a has been taken from state s and every state-action pair is visited an unbounded number of times and both $\sum_{k=0}^{\infty} \alpha_k(s,a) = \infty$ and $\sum_{k=0}^{\infty} \alpha_k^2(s,a) < \infty$, hold true, the Q-values will converge to the limit.

4 Parameter Tuning

5 Conclusion