CS229 Milestone Report
Reinforcement Learning to Play Mario

Yizheng Liao, Zhe Yang, Kun Yi

November 16, 2012

1 Introduction

Using artificial intelligence (AI) and machine learning (ML) algorithms to play computer games has been widely discussed and investigated, since valuable observations can be made on the machine learned play pattern vs. that of the human player, and provide knowledge on how to improve the ML algorithms. Mario AI Competition [1] is such a competition that uses AI and ML techniques to play the classic title Super Mario Bros. The Competition provides the evaluation system and benchmarks for the competitors every year.

Reinforcement Learning (RL) [2] is one of the widely studied methods in ML and is one of the promising approaches to implement ML game agents that can simulate the behavior of a player [3]. In this project, we are studying how to construct a Mario controller agent, which can learn from the game environment, by using RL method. One of the difficulties of using RL method is how to define state, action, and reward. We solve this problem by simplifying the environment variables, which are provided by the evaltion system. Another issue is how to implement the agent efficiently since the computer game playing requires real-time response. To solve it, we use Q-Learning algorithm to online evolve the decision strategy that aims to maximize the reward. Our controller agent is trained and tested by the 2012 Mario AI Competition evaluation system. The results are compared with the Competition benchmarks.

The rest of this report is organized as follow: Section 2 provides an overview of Q-learning algorithm and the Mario AI interfaces; Section 3 explains our Q-learning algorithm and how we define the environment variables; Section 4 gives the details for future works.

2 Background

In this section, we will introduce the Q-learning algorithm and the Mario AI interfaces.

2.1 Q-Learning

Q-learning treats the learning environment as a state machine. Q-learning collects several environment variables and define them as some states. The learning process always on some states in Q-learning algorithm. Each state is assigned a unique state number. Q-value is computed for each pair of state and action and is saved in Q-table. Here, $Q(s, a)$ denotes the Q-value, where s is the state and a is the action. For each action in a particular state, a reward will be given. The Q-value is updated based on reward by following rule:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha(r + \gamma \max(Q(s_{t+1}, a_{t+1}))) \quad (1)$$

In the Q-learning algorithm, there are four main factors: current state, chosen action, reward and future state. In (1), $Q(s_t, a_t)$ denotes the Q-value of current state and $Q(s_{t+1}, a_{t+1})$ denotes the Q-value of future state. $a \in [0, 1]$ is the learning rate, $\gamma\,[0, 1]$ is the discount rate, and r is the reward. (1) shows that for each current state, we update the Q-value based on the maximum Q-value of the next state.

2.2 Mario AI Interface

Super Mario Bros is a very famous computer game. The game is controlled by six keys: UP, DOWN, LEFT, RIGHT, JUMP, and SPEED. In the interface, UP key is not used. Therefore, there are totally $2^5 - 8 = 24$ possible actions (we remove the conflicting keys).

Mario in the game has three statuses: small, big, and fire. In fire mode, Mario can shoot fireball to beat enemies. A small Mario can eat mushroom to become a large Mario and a large Mario can eat flower to become a fire Mario. The Mario will lose the status if it is affected by an enemy. When a small Mario is attacked by an enemy, the game is over. The enemies include Goomba, Koopa, Spiky, Bullet Bill, and Piranha Plant. The enemies can be eliminated by fireball or jumping to it. For the specification of each enemy, please see [3] and [1] for details.

There are some items in the environment that can give Mario reward. Mushroom and flower are power-up items and also give reward. Coin and bricks can also give reward.

There are four landforms and obstacles. They are hill, gap, cannon, and pipe. Mario should adopt different actions for each landform.

The evolution system is programmed by Java. It provides several functions for the developers to check environment and return the environment variable arrays, which contain the information about item and enemy within a defined grid. The frequency of benchmark is 24 frames per second. The environment checking functions are called every frame. Therefore, the Q-function algorithm has to be completed within 42 milliseconds.

3 Mario Controller Design Using Q-Learning

Each frame can be decomposed into several blocks. The size of each item is equal or larger than one block size. As discussed above, there are 19 possibility for each block. Now if we take a 22×22 grid, there are totally

$$19^{22\times 22-1} \times 24 \approx 10^{619} \tag{2}$$

state-action pairs. Then we will raise memory issue and computational issue. Therefore, we have to re-define states to simplify the problem. In this project, we define two state-action spaces.

In the first state-action space, Rule 1, we choose 10 states.

- Mario mode: 0 - small, 1 - big, 2-fire
- Higher enemies near front side: weather or not there is an enemy in the right side of Mario within a given range and its position is higher than Mario.
- Higher enemies near backside: weather or not there is an enemy in the left side of Mario within a given range and its position is higher than Mario.
- Lower enemies near front side: weather or not there is an enemy in the right side of Mario within a given range and its position is lower than or equal to Mario.

- Lower enemies near backside: weather or not there is an enemy in the left side of Mario within a given range and its position is lower than or equal to Mario.
- Brick exist: 1 - there exist a brick within a given range; 0 – there does not exist a brick within a given range
- May upgrade: weather or not there exists mushroom or flower within a given range
- Near gap: weather or not there is a gap within a given range
- On gap: weather or not Mario is jumping across a gap or falling in the gap
- Enemy type: represent the enemy type within a given range

In addition, we choose six actions.

- Stomp attack subtask: attacking enemies by stomping
- Fireball attack subtask: attacking enemies by fireball
- Shell attack subtask: attacking enemies by shell
- Upgrade task: trying to upgrade Mario if possible
- Pass through task: controlling Mario to cross the landforms and obstacles to move rightward
- Avoidance task: avoiding enemies and not getting hurt

Now, we reduce the number of state-action pairs to $3 * 2^8 * 4 * 6 = 18432$.

In the second state-action space, Rule 2, we reduce the last state in Rule 1. Now we will use the same rule for each enemy. Therefore, now the first three actions are the same. Hence, we have $3 * 2^8 * 4 = 3072$ possible state-action pairs.

4 Next Step

For the rest of this project, we are going to train the Q-table based on the algorithms given above. Then we will compare our results with the benchmark. In addition, instead using self-defined reward rule, we will use the reward rule given by the evaluation system. Further, rather than updating Q-value every frame, we will update it every two or five frames. In this case, we relax the constraint on computational complexity. It allows us to implement a more complicated state-action space.

References

[1] J. Togelius, S. Karakovskiy, and R. Baumgarten, “The 2009 mario ai competition,” in *Proceedings of the IEEE Congress on Evolutionary Computation*, Citeseer, 2010.

[2] R. Sutton and A. Barto, *Reinforcement learning: An introduction*, vol. 1. Cambridge Univ Press, 1998.

[3] J. Tsay, C. Chen, and J. Hsu, “Evolving intelligent mario controller by reinforcement learning,” in *Technologies and Applications of Artificial Intelligence (TAAI), 2011 International Conference on*, pp. 266–272, IEEE, 2011.