# Super Mario Bros. Meets Reinforcement Learning

CS 182 Final Project Proposal

Gabe Grand and Kevin Loughlin

October 28, 2016

## 1 Project Statement

Super Mario Bros. (SMB) is a classic platform video game released by Nintendo in 1985. In SMB, the player must navigate Mario through a series of increasingly-difficult levels, collecting coins and avoiding obstacles along the way. Our goal is to use reinforcement learning to create an AI agent that is proficient at SMB.

From an AI perspective, SMB poses several interesting challenges. First, Mario encounters a wide variety of obstacles, enemies, and terrain within and across the 32 levels, so the state space is vast. Second, many elements of the state are not observable, or are only partially observable. For instance, platforms, coins, and flagpoles are all labeled as "object" in the state representation (more detail in the next section). Finally, there exists no single optimal policy, since the game has several interrelated but distinct goals. Mario must reach the end of the level as quickly as possible, but various actions, like collecting coins and defeating enemies, can contribute to the score.

## 2 Framework and State Representation

In order to focus our development efforts on creating algorithms to control the behavior of Mario, we will utilize an existing framework to provide certain elements of the model, such as the game state, reward function, and emulation of the game itself. The Open AI Gym is a toolkit for developing reinforcement learning algorithms in Python [2]. It contains various environments, ranging from classic video games to 3D physical control systems.

For our project, will work off of the Gym Super Mario environment bundle. The bundle provides all 32 levels of SMB in two state representations. The first is a 256 x 224 array of RGB values representing the raw pixel output of the game. The second is a 16 x 13 array of square tiles, which represent the following game elements:

- 0: empty space
- 1: object (e.g. platform, coins, flagpole, etc.)
- 2: enemy
- 3: Mario

Achieving proficiency of control using only the raw pixel values is an open research problem, whose solution generally requires the use of computationally-expensive methods like neural networks[9]. In order to reduce training costs and avoid unnecessary complexity, we will work off of the tiled state representation, rather than the pixel-based one.

## 3 Algorithms and Related Course Topics

Q-learning, as covered in class, is a reinforcement learning (RL) algorithm that involves determining the "Q" values of each state-action pair in a state space through sampling. The key point is that, unlike Q-value iteration, we will not know the transition model for the state space. Thus, we will need to determine our policy by updating $Q$ values until they converge, done via through the following pseudo code provided in Lecture 10:

Initialize each $Q(s, a)$ arbitrarily (often to 0 for simplicity)
For each episode:
    Initialize $s$
    While $s$ is not terminal:
        Choose $a$ from $s$ using $\epsilon$-greedy policy derived from $Q$
        Take action $a$, observe reward $r$ and new state $s'$
        $Q(s, a) = Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
        $s \leftarrow s'$

As for Partially Observable Markov Decision Processes (POMDP's), they are somewhat similar to Hidden Markov Models (HMM's) discussed in class. However, the key difference is that we can control the state transitions for POMDP's. POMDP's are essentially solved with value iteration. The main issue is that discrete POMDP's (which is what we are dealing with) are equivalently represented as continuous MDP's. In value iteration, we're required to loop over the next possible states, but in a continuous state space, there are uncountably infinite next states. Therefore, the difficulty that lays ahead for us is understand how to modify the algorithm such that handle continuous state spaces.

## 4 Expectations for System Performance

Q-learning is a model-free method that is agnostic of the underlying structure of the problem. In contrast, POMDP is a model-based method that assumes the agent cannot directly observe certain elements of the MDP state. The Mario world contains rich and consistent structure. For instance, in nearly all levels, the lower-half of the screen is occupied by the ground, while the upper-half is largely open and contains rewards like coins and power-ups. For this reason, we expect the POMDP model to outperform the Q-learning model.

In both cases, we expect that our algorithm's performance will be noticeably worse than that of an average human player. This is because SMB is a member of a class of games that require building causal, compositional, and symbolic models of the world, rather than simply solving pattern recognition problems [6]. For instance, a 10-year-old playing SMB for the first time can draw on his or her existing knowledge of intuitive physics to infer Mario's trajectory when jumping from

one platform to another. Reinforcement learning models posses no such intuition.

## 5 Division of Labor

To begin, we will need to read through the various papers cited at the end of this proposal. We plan to divide up the reading load 50/50. If there is a particularly important paper in either of our sets, we will have the other team member read it; otherwise, we will simply provide a summary.

We expect that the Q-learning portion of the project will not be too difficult, as we coded similar algorithms in class as part of the Pacman problem sets. As such, the algorithmic design and coding will be shared between the two of us, just as we did on our Q-learning problem set.

When we move into the POMDP portion of the project, which we have not covered in class, we will likely need to subdivide the workload into sections for each of us to complete. Since the model we choose is the most important aspect of our success, we will establish the parameters of our POMDP together. Once these are decided, one of us will focus on coding the exploration side of the POMDP, while the other will work to establish the behavior upon convergence.

Finally, it makes sense that we will handle the portions of the write-up more relevant to our own tasks. We each plan to host our project results on our personal websites, and will necessarily share the work of preparing our final presentation.

## 6 List of Resources

The Open AI Gym, including related documentation, is the main resource that we will draw on for our project [2]. Within the Gym framework, we will use the Gym Super Mario environment. Additionally, the environment interfaces with the open-source FCEUX Nintendo Entertainment System emulator to render the game. For our algorithms, we will draw on papers from the reinforcement learning and POMDP literature. A preliminary list of sources is provided below.

## References


[1] Tomás Banzas Illa. Reinforcement learning in an emulated NES environment. 2015.

[2] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[3] Anthony Cassandra, Michael L Littman, and Nevin L Zhang. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 54–61. Morgan Kaufmann Publishers Inc., 1997.

[4] Anthony R Cassandra, Leslie Pack Kaelbling, and Michael L Littman. Acting optimally in partially observable stochastic domains. In *AAAI*, volume 94, pages 1023–1028, 1994.

[5] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[6] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *arXiv preprint arXiv:1604.00289*, 2016.

[7] Yizheng Liao, Kun Yi, and Zhe Yang. CS229 final report: reinforcement learning to play Mario. Technical report, Technical report, Stanford University, 2012.

[8] T Michael and I Jordan. Reinforcement learning algorithm for partially observable markov decision problems. *Proceedings of the Advances in Neural Information Processing Systems*, pages 345–352, 1995.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[10] Shiwali Mohan and John E Laird. Relational reinforcement learning in infinite Mario. *arXiv preprint arXiv:1202.6386*, 2012.

[11] Nicholas Roy, Geoffrey J Gordon, and Sebastian Thrun. Finding approximate pomdp solutions through belief compression. *J. Artif. Intell. Res.(JAIR)*, 23:1–40, 2005.

[12] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River, 2003.