Practical 4 - Reinforcement Learning with Swingy Monkey

Tyler Tarsi, Darrin Gilkerson, Jeff Balkanski
tylertarsi@college.harvard.edu, dgilkerson@college.harvard.edu, jbalkanski@college.harvard.edu

May 7, 2019

1 Technical Approach

As detailed in the practical handout, the main challenges were the (effectively) continuous state space, the unknown dynamics of the system, and the stochastic gravity component. To handle the unknown dynamics, we would have to use RL, rather than a standard MDP solving approach. To handle the stochastic gravity component, we had our learner infer the gravity within the first 3 ticks of each epoch. To handle the continuous state space, we discretized the state space into bins, which we then optimized for our learner. And to handle the exploration-exploitation trade-off, we explored the ϵ-greedy algorithm.

Discretize Bins Approach:
The state space consists in the distance from the monkey to tree, height of top and bottom of tree gap, height of top and bottom of monkey, and the velocity of the monkey. We recognized that the more states we had, the longer our learner would take to optimize, so our first priority was in reducing these. We decided to use take the state space of 6 parameters and reduce it to 2: the vertical distance between the top of the monkey and the top of the tree gap (we were able to ignore the bottom gap between monkey and tree since the tree gap is constant), the horizontal distance between monkey and tree, the monkey's velocity, and the distance from the bottom of the monkey to the bottom of the screen (once again ignoring a metric, distance from top of monkey to top of screen, since the screen height is constant).

Once we reduced our parameter space to 4, we discretized each parameter into bins. We did this through a process as follows: int(np.round(float(Actual State Value) / Appropriate Division)). By dividing by an "Appropriate Value" and rounding to the nearest int, we discretized what was continuous into sets s where $2 < |s| < 37$. The appropriate values were determined by trial and error, and by exploring the question: how important is this parameter to our learner? How much variance should in this parameter should our learner be concerned with? Using this approach, we settled on 4 appropriate values.

Our state space having been properly reduced, we could proceed to the actual learning. For this, we used an 100% exploitative off-policy approach (with the Q-learning update) such

that:

$$\frac{\partial \mathcal{L}}{\partial w_{s,a}} \approx Q(s,a;\mathbf{w}) - [r + \gamma \max_{a' \in A} Q(s',a',\mathbf{w})]$$

and to update

$$w_{s,a} \leftarrow w_{s,a} - \eta \frac{\partial \mathcal{L}}{\partial w_{s,a}}$$

The final task was to infer the gravity. We wrote a simple function which involved tracking the monkey's top height across ticks, ensuring that no action was taken (because jumping would complicate things), and seeing how quickly the height changes. For the first 3 ticks of each epoch, the monkey updates Q values in which the gravity is set to None in the state space. After the first 3 ticks, the gravity is inferred and input into the state space. From then on, after every epoch change, the learner is updating the appropriate Q values and accounting for gravity properly within 3 ticks.

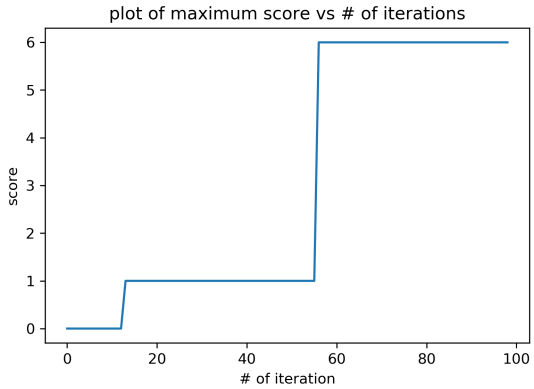
Epsilon-Greedy Approach:
To confront the exploitation-exploration trade-off, we also wanted to implement an ϵ-greedy approach and see how that compares to a strictly exploitative approach. In the exploitative approach, when we are in state s, we can simply take the action $a = argmax_{a \in A} Q(s;a)$ based on our current estimate of the Q-function. In an exploratory approach, we might select actions at random, encouraging the Q-function to to discover values for many state-action pairs and potentially finding better actions. We want to balance these two - exploitation lets us arrive at the optimal policy faster, but exploration is necessary to properly understand the state space. One popular approach is called ϵ-greedy, where we take $a = argmax_{a \in A} Q(s;a)$ with probability $1 - \epsilon$ and pick $a \in A$ uniformly at random with probability ϵ. This is the strategy that we chose to implement because of its exploitation-exploration balance and its practical usefulness found in the literature.

2 Results

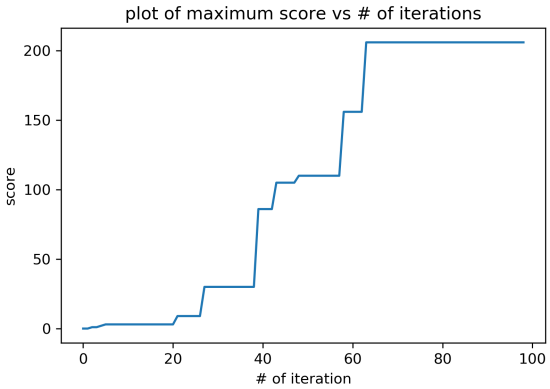
Model	Best Score (100 epochs)	# Epochs to score 100
1/ Binary Features	6	None (best score is 6)
2/ Bucketed Features	206	43
3/ Inferred Gravity, Bucketed Features	2273	49
4/ Inf. Gravity, Bucketed Features, ϵ-greedy	130	103

Table 1: Each of the models we iterated on and their corresponding metrics.

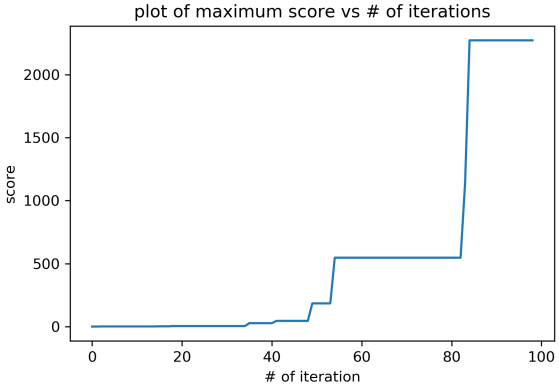
We run the last model for 200 rounds to have better grounds of comparisons. We chose to use the following metrics to analyze the quality of our models:



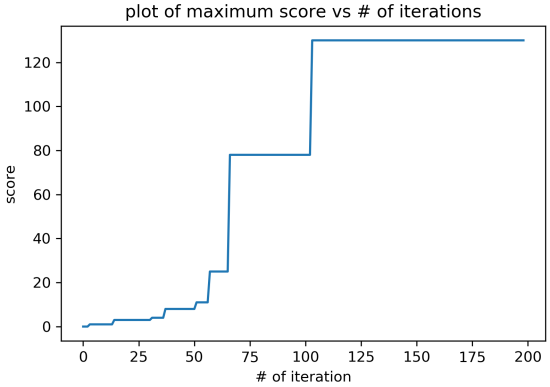
(a) model 1



(b) model 2



(a) model 3



(b) model 4

- **Best score after 100 epochs** - this allowed us to see, overall, how well the model learned to play the game. By 100 epochs, our models were at a point where their improvements were mostly marginal. So this allows us to see how well each model learned the Q-values.
- **# Epochs it took to score 100** - This tells us something different than the first metric. Even if a model learns the Q function very well, how long does it take to do that? Ideally, learning faster is better, but we certainly face a trade off in terms of model complexity and learning time.

We believe using these 2 metrics together gives us an accurate picture of the quality of each model.

3 Discussion

We began our modeling process by defining the feature space. We chose to use the following features:

- **Monkey Velocity (MV)** - We included this feature because they velocity changed each epoch and plays a significant role in the monkey's decisions.
- **Monkey Distance to bottom of screen (M2B)** - We included this because there is a large penalty for the monkey falling off of the bottom, so if it is too close it should elect to jump.
- **Monkey to Tree horizontal distance (M2Th)** - We included this because the monkey's horizontal distance to the tree is a crucial factor in deciding when to jump.
- **Monkey distance to top of tree (M2Tv)** - It's important to include this because it defines where the monkey is in reference to the opening of the tree.

Our first model included each of these features as a binary. For example, M2Th distance was either below or above 150 pixels, MV was either above or below 15 (the average). There were 2 options for each of the 4 features, giving us a state space of $2^4 = 16$ states. The binary feature model did not perform very well (see above), and we believe this is because the binary features were not descriptive enough of the state space.

Our next model expanded on the binary features to include more buckets. We experimented with different bucket sizes for each of the same features and empirically found that MV and M2B were actually well representative of the state space as binary features. M2Th was optimally bucketed into bins of size 150, and M2Tv was optimally bucketed into bins of size 33. This model ran with much better success than the binary one, as a result of its expanded state space.

Using this new state space, we realized that each epoch's gravity was different (either 1 or 4). By inferring the gravity during the first few timesteps of each epoch, we believed we could achieve much greater performance. We implemented a function that observes the monkey's change in height at the beginning of each epoch and then sets the learner's gravity to either 1 or 4 based on the information inferred. This led to significantly better performance (see table above).

Finally, we wanted to explore the ϵ-greedy approach to the exploitation-exploration trade-off. Previously, we had been using an entirely exploitative approach, but using ϵ-greedy allowed our learner to learn the state-action value function more wholly. It trained much slower, because we were deliberately choosing potentially sub-optimal actions. But it did converge to a solid value function eventually. It's best score was not great after 100 epochs, and this makes sense because it takes much longer to train.