Managing Luck: A Multi-Armed Bandits Meta-Agent for the Angry Birds Competition

Arnaud Jutzeler, Mirko Katanic and Jason Jingshi Li
EPFL
Lausanne, Switzerland
jason.li@gmx.ch

Abstract

One of the problems we have encountered to play Angry Birds in a competition setting is that there are different strategies that work well on different levels, and one has to be lucky in finding a basic strategy that matches well with the given level. When we have many different levels to play, there is an exploration/exploitation tradeoff between different strategies. The trade-off in such a setting can be interpreted as a multi-armed bandit (MAB) problem, which has been extensively studied in Machine Learning literature. For the Angry Birds Competition in 2013, we implemented the "Beau-Rivage agent", a meta-agent that allows us to choose the next level and its best strategy.

1 Introduction

The Angry Birds is a addictive game originally designed for mobile phones, where different birds launch themselves towards pigs hidden behind structures made of wood, ice and bricks. The first Angry Bird AI competition was ran in 2012 at the Australian AI Conference, where agents are programmed to control target and trajectory of the birds from a screen shot. In that competition, the winner turned out to be the *Naïve agent*, where the agent aim directly at the pigs, without considering what obstacles lie in the way.

The game can be interpreted as search tree with a very shallow depth d, but a large branching factor. More importantly, without the actual physics engine, it is difficult to estimate the consequence of the actions. For a human player, there are many different strategies to play a given level. One could target directly at the pigs, trying to topple over some structure, undermine a structure from below, target some difficult blocks to clear the way for the next shot, or possibly many others. The odds are that the optimal strategy would be different for different levels, and there is no one static strategy to solve all possible levels without replicating the physics engine and solve the game via search.

Given the uncertainties arise from the Computer Vision module and many randomness built into our strategies, in the Angry Birds AI Competition setting repeating the same strategy on the same level would offer different outcomes. Therefore, it is more reasonable to model the outcome (score) of a given strategy-level pair as a distribution. Playing a level with a certain static strategy would then be analogous to sampling from the corresponding distribution.

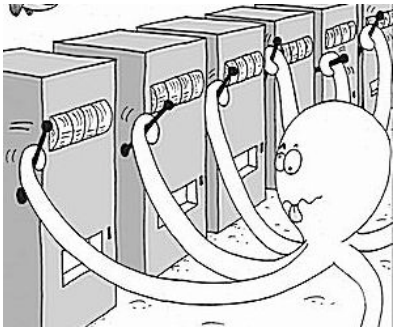

Figure 1: A illustration of the multi-arm bandit problem (image courtesy of Microsoft Research)

The problem of optimizing sampling from different distributions is known as the Multi-Armed Bandit (MAB) problem, and it has been studied extensively in Machine Learning. In this paper, we describe our game-playing agent as a meta-agent trying to match different strategies to different levels.

2 The Strategies

We have implemented the following basic strategies for the Angry Birds Competition 2013. At any one time, the MAB meta-agent would select one of these strategies to play a given level. A more comprehensive evaluation of the individual strategies can be found at [Katanic, 2013].

2.1 Naïve Strategy

The first strategy is the naïve strategy, where the agent selects the pigs as the targets. It was the winner of the previous Angry Birds AI Competition in 2012, and part of the platform offered by the competition organizers. We made further modifications by implementing a new hit checker in order to alter the trajectory and avoid hitting into background objects. We also modified the tapping function for handle birds of different colors.

Figure 2: The TallWide strategy: factoring for tall (left) and wide (right) structures.

2.2 LowKick Strategy

One strategy we have found to be fairly effective in practice is modifying the naïve agent such that we target the block sitting just below the pigs. We termed this "LowKick" strategy, as it is designed to undermine the structure supporting the pigs.

2.3 TallWide Strategy

The TallWide strategy aims to exploit the structures of the blocks. It is reactive agent that is choosing its target each turn by assigning preference values to all objects present in the game. The heuristics used by the agent is based on observing connected sets of game blocks. Its basic idea is that tall structures can be destroyed by hitting them on the top and tipping them over, and wide structures can be destroyed by undermining their foundations. Empirically, we have found that it is an effective strategy on many benchmark levels.

2.4 ClearTheWay Strategy

The ClearTheWay strategy is a modification to the TallWide strategy that will also consider future possibilities. It is following the same structure and heuristics of TallWide, with two important changes. First change represent agent looking in the future. When identifying color of the active bird, it is also looking for the type of the bird that will be used in a next turn. Preferences for blocks are now further reduced if the bird that will be used in next shot is good against the block and current one isn't. Reason for this is that in this way agent will avoid blocks that it can destroy more easily in next turn, so it does not waste current shot.

3 The MAB Meta-Agent

In the competition setting, we have to match the strategies to different game levels in a finite amount of time. Therefore in general, given a set of N levels and a set of M strategies, we consider a $N \times M$ arm bandit, where each arm is an element of $N \times M$. We use the following algorithm known as Upper Confidence Bound (UCB) [Auer *et al.*, 2002] to choose the next arm (level-strategy pair) to play.

- Initialization: play each arm once.
- Loop: play the j^{th} arm that maximizes $\mu_j + \sqrt{\frac{2ln(n)}{n_j}}$, where
  - μ_j is the average reward from the j^{th} arm
  - n is the total number of plays so far
  - n_j is the total number of plays involving the j^{th} arm

We make a further assumption that it is more likely to gain more points by passing a new level than replaying an existing level. Therefore, for the competition we created a two-tiered strategy by first we try to pass all the levels, and then trying to gain more points by playing levels that we have already passed.

3.1 Passing the levels

In the first stage, we choose the reward to be the amount of points gained at the end of play of a level, even though they are not added to the total competition score if the level was not passed (all pigs killed). This gives us an indication of how well we are doing with a strategy at a given level. The initialization is done by playing all the levels with each strategy, but once a level is passed, we remove all the arms associated with the level.

3.2 Improving on existing score

Once all the levels are passed, we then look at improving the existing score. Here, the reward is interpreted as the amount of gain from the existing score. When the gain is negative, we set the reward to zero. The UCB algorithm would continually select the next level/strategy pair until the end of the allocated time set by the competition.

Acknowledgments

We like to thank the competition organizers for providing the software platform for our agent and organize the Angry Birds AI Competition at IJCAI'13.

References

[Auer *et al.*, 2002] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[Katanic, 2013] Mirko Katanic. Semester project: The angry birds project. Technical report, Artificial Intelligence Laboratory, EPFL, July 2013.