Tackling an Optimal Stopping Problem with Q-Learning

Zachary Beisler

Vasant Honavar – Data Sciences 497

The Pennsylvania State University

2 May 2019

**Optimal Stopping Problems**

Optimal stopping problems are choice-based games, limited by the constraint that potential options may only be viewed sequentially and that once an option has been viewed and passed up it cannot be chosen at a later time. The particular problem being examined in this paper involves a sequence of positive integer values, bounded above by a constant that is known a priori to the guesser. The goal of the problem is to choose the option with the highest value, without being able to look at all of the values simultaneously (as per the constraints on the problem). The common name for this type of optimal stopping game is “The Secretary Problem”, referring to a situation in which candidates are sequentially interviewed for a secretary position and an employer stands to save time by stopping interviews early.

The problem was first proposed in publication form by Martin Gardner in a 1960 column in *Scientific American*, where he referred to it as “the game of googol” – a variant in which the upper bound on the numbers was one googol (). When Gardner published the problem he did not have a mathematical solution, although shortly after being contacted by Gardner, Leo Moser and J. R. Pounder were able to publish a correct analysis.

**Solution to the Maximum-Value Formulation**

The maximum-value formulation of the problem can be stated as follows:

Given a sequence of applicants , the goal of the game is to choose the applicant such that is the *best* candidate. The game is played by making sequential choices, one for each candidate, under the constraint that the choice for each candidate *must* be made immediately after considering them. The only two possible choices are and , wherein once an applicant is accepted the game ends and once an applicant is rejected the next applicant is considered. The only information after considering candidate is if is maximal so far, i.e. if

The solution to this problem is a stopping rule, where the chooser rejects the first applicants and then accepts the first maximal candidate encountered afterward. A proof that such strategies are optimal was provided in 1984 by Thomas Bruss. An easier to follow yet less rigorous proof of the same result can be found on theWikipedia page for the secretary problem, and it looks like this (Proof Taken from the Wikipedia page):

As , this sum can be approximated by an integral using the substitutions , , and .

The final step is to optimize this expression with respect to and find a maximum, which can be done by setting its derivative to . The final result is that:

Which gives an approximation to the optimal strategy, where the first applicants should be ignored, and the next best taken.

**The Cardinal Payoff Variant**

The specific variant of the secretary problem investigated in this paper is the so-called cardinal payoff variant, where the goal is not to choose the maximal applicant, but instead to maximize the expected value of the chosen applicant. Each applicant’s true value is drawn independently from a uniform distribution, and the rules for selecting an applicant are the same as they were in the original formulation.

The optimal cutoff strategy in this case is to reject the first applicants, as shown by **Bearden** in 2006. A proof of this result will not be given here. This optimality will turn out to be slightly problematic for implementing a reinforcement learning algorithm to cope with, as its dependence on limits how well an agent can generalize its policy to deal with stopping problems with varying lengths.

**Reinforcement Learning Approaches**

The motivation for examining stopping problems with reinforcement learning algorithms is to evaluate their ability to learn in a stochastic and long-term planning oriented environment. Even though stopping problems of the variety looked at in this paper have been solved closed-form, they can serve as good indicators of how current learning algorithms cope with environments that require long-term planning and consideration for success. Indeed, upon first learning of such stopping problems it is not obvious that a better-than-chance solution would even exist. A general intelligence, the ultimate goal of the field of reinforcement learning, should be able to learn the solutions to stopping problems.

The stopping problems outlined could be seen as construed and abstract, but I argue that many real-world problems have similar structure – consider browsing internet search results, for example. An agent designed to find the best articles using internet searches would have to balance time spent evaluating potential links with the perceived optimality of links it has seen. This can be seen as a slightly less constrained stopping problem, where the agent is allowed to double back and select previously evaluated links. Any problem that has an element of timing to it (best time to execute an action in a time window) would also be stopping problem. A concrete example problem is when to sell stocks, should an agent sell their stocks at the first instance of a maximum, or should it be “greedy” and wait banking on the stock value to trend even higher? This is all just to show that making sure learning algorithms can solve optimal stopping problems is not just an abstract exercise, but that it can have applications in both real-world problems and future algorithm designs.

**Q-Learning**

All reinforcement agents in this paper were developed using variations on the popular Q-Learning algorithm introduced by Watkins in 1989. The general approach is to store the policy function of the learner in a structure known as a q-table, with q-values assigned to each possible action the agent could take at each state. Q-values represent the estimated future reward the agent can expect by taking an action, and larger q-values correspond to actions the agent believes are better. Other reinforcement learning algorithms exists, and it is encouraged for future work to evaluate them on optimal stopping problems similar to the ones in this paper.

**The Environment**

The environment that the agents will learn in is similar the formulation for the cardinal-payoff variant of the secretary problem outlined previously. The agents will progress through a list of integers independently sampled from some uniform distribution, with the option to accept and reject at each step. Upon accepting, the agent will receive the accepted value as reward. Because the environments are randomly sampled with high variability, training was limited to distinct environments, which improved convergence to solutions much better. The generalization of the agents’ learned policies was tested on independent stopping problems and will be discussed later.

*The optimal stopping problem environment. The are sampled from a uniform distribution.*

The naïve way to implement a q-table would be to have states, where is the length of the stopping problem and is the upper bound. Each state would have q-values associated with it, corresponding to the actions of accepting or rejecting the current value. This approach is not practical, however, as the upper bound can be arbitrarily large – for example a variant mentioned earlier had . This would not only be infeasible due to memory constraints, it would also make learning take prohibitively long as in order to fully understand the environment the agent would have to have seen each possible value at each possible position, and even if each value was guaranteed to appear once in each problem (which is not generally the case) that results in a minimum of possible environments to explore. Another problem with the hard-coded q-table approach is that the agents learned policy would generalize poorly, if at all, to environments with different and values then those that were trained on.

In order to promote the agents learning of general heuristics, quantization of the state space was employed. The q-table was set to have a fixed size based on a new parameter , the step precision, equal to . States are bucketed by how far along they occur in the overall stopping problem, with the th state falling into a bucket labeled by , rounded to the nearest multiple of . For example, a step precision of results in buckets being created. Two separate approaches to dealing with this same problem with regard to the values were employed, and they along with how well they performed will be discussed in the upcoming sections on the particular agents developed for the problem.

**Value-Oriented Agent**

The value-oriented agent had its q-table implemented by simply repeating the quantization process used on the steps – but on the value at each step. The value precision, then, is denoted by . This way, at each state, the agent would be exposed to two percentages telling it both how far along the problem it was, and what fraction of the upper bound the current value was. This is technically not allowed as per the original rules of the stopping problem, however it seemed like a natural choice for reducing the state space. As such the q-tables for these agents had dimensions . Several value agents were trained on stopping problems varying in length and upper bound, and then were evaluated against the stopping rule as a baseline optimality check. Averaging a reward around of the upper bound on a stopping problem is equivalent to choosing randomly. For all of the value agents, training steps were used with a learning rate of and a discount factor set to . The evaluation of the agents was done on independent stopping problems of length and upper bound .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Problem Length** | **Upper Bound** | **Step Precision** | **Value Precision** | **Average Reward** |
| 10 | 100 | 0.1 | 0.01 | 55.75% |
| 100 | 100 | 0.01 | 0.01 | 50.01% |
| 100 | 1000 | 0.01 | 0.01 | 50.01% |
| 1000 | 100 | 0.01 | 0.01 | 51.73% |
| 1000 | 1000 | 0.01 | 0.01 | 51.88% |
| Optimal Heuristic: Stopping Rule | | | | 79.65% |

*Table of average reward values as a fraction of the upper bound for various value agents. The agents were barely able to perform above random chance.*

As can be seen from the table of values above, the value-oriented agent approach was essentially a failure, with their average rewards about equivalent to picking a stopping point at random. It is clear that quantizing the value states doesn’t work well for the problem – which makes sense intuitively. The actual values encountered shouldn’t play a role where the optimal cutoff point is, and the formulaic solutions presented earlier back that notion. By having such large q-tables, the agents likely also did not get a chance to fully explore the state space.

**Rank Based Agents**

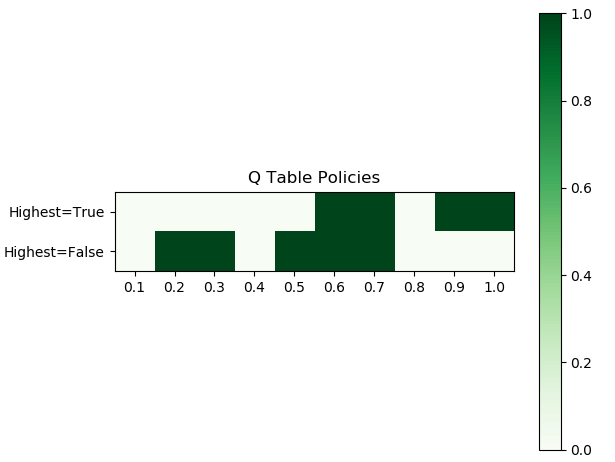
Instead of value quantization, which evidently does not fit the problem well, another method of reducing state space is to keep the steps quantized but only disclose to the agent if their value is so-far maximal. That is, to stay true to the original problem where this was the only piece of information available. We will call this implementation a rank based agent.

For a rank based agent, each state is parameterized by a quantized step value and a Boolean value indicating whether the current value is so-far maximal. This reduces the dimensions of the q-table to , a marked improvement in memory usage over the value-oriented agents. As with the value-oriented agents, several rank based agents were created, trained, and evaluated against the optimal baseline. The same number of training steps, learning rate value, and discount factor were used with the rank based agents.

|  |  |  |  |
| --- | --- | --- | --- |
| **Problem Length** | **Upper Bound** | **Step Precision** | **Average Reward** |
| 10 | 100 | 0.1 | 81.31% |
| 100 | 100 | 0.01 | 58.65% |
| 100 | 1000 | 0.01 | 51.14% |
| 1000 | 100 | 0.01 | 49.89% |
| 1000 | 1000 | 0.01 | 49.89% |
| Optimal Heuristic: Stopping Rule | | | 80.36% |

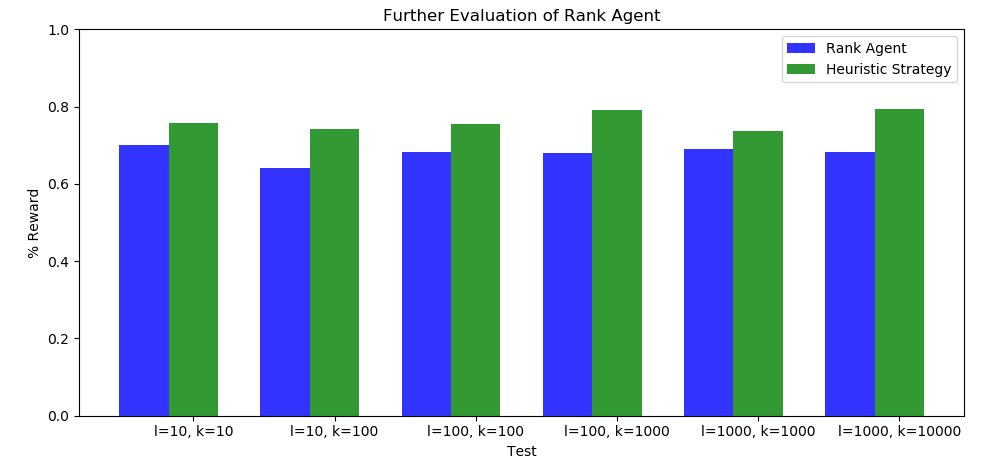
*Table of average reward values as a fraction of the upper bound for various rank based agents. Notice that the agent that trained on smaller values managed to perform as well as the optimal heuristic.*

Rank based agents trained on larger problems faired just as poorly as the value-oriented agents, but the agent trained on the smallest problems was able to meet the optimal heuristic; even exceeding it on the testing set of problems. This is surprising because the evaluation problems were all of length and upper bound , x the parameters that the agent was trained on. This seems to suggest that the rank-based approach is robust to changes in the superficial qualities of general stopping problems. The q-table for the agent trained on and is visualized below.



*Visualization of the q-table for the rank-based agent trained on small stopping problems. The value at each cell is policy, when the agent accepts and 0 when the agent rejects.*

Looking at the q-table, one can see that the agent doesn’t seem to be learning the behavior we want it to at all. For example, it is choosing to accept values that aren’t even so-far maximal at step percentages of and . This is in direct contrast with the mathematical optimal stopping rules, where the chooser should only ever accept when a value is so-far maximal after a certain cutoff. This particular well performing agent was re-evaluated, this time on a larger set of differing stopping problems.



*Results of further evaluation of the rank agent trained on smaller stopping problems. The agent is consistently better than chance but not as effective as the heuristic.*

Even though the q-table seems sub-optimal at best, this agent still performs very well and very consistently on larger stopping problems, even one with integers and upper bound . It seems that the policy the agent learned is still effective, even if it isn’t perfect.

**Closing Comments**

Optimal stopping problems, even though they are solved, are still interesting problems for the development and evaluation of reinforcement learning algorithms. More sophisticated approaches, such as Deep-Q learning, may provide better results – and might even be able to match the mathematically optimal strategies in performance. The effectiveness of using smaller, simpler environments in conjunction with general state spaces gives insight into the value of agents being able to generalize policies to tackle similar but non-identical environments.

**Works Cited**

Bruss, F. Thomas. “Sum the Odds to One and Stop.” *The Annals of Probability*, vol. 28, no. 3, 2000, pp. 1384–1391., doi:10.1214/aop/1019160340.

“Secretary Problem.” *Wikipedia*, Wikimedia Foundation, 25 Apr. 2019, en.wikipedia.org/wiki/Secretary\_problem.

Watkins, Christopher John Cornish Hellaby. "Learning from delayed rewards." (1989).