The Secretary Problem: Learning with Experts and Oracles

Sikata B. Sengupta
Departments of Economics and Mathematics
Stanford University
Stanford, CA 94305
sikata@stanford.edu

Alain Schläpfer
Department of Political Science
Stanford University
Stanford, CA 94305
alainsch@stanford.edu

Abstract

Threshold-based strategies for optimal stopping problems have long been studied within a wide range of fields. The secretary problem is a well-known example of such a problem where one can derive the optimal policy for an agent faced with choosing the best among N candidates by making sequential accept/reject decisions. Here, we consider a multi-round version of the secretary game, a problem to which reinforcement learning methods have been applied successfully. In this paper, we study imitation learning for such games, with teachers with different degrees of knowledge available to them. We explore the consequence of the mismatch between the teacher's observation/state space and that of the agent on the ultimate performance after training. We propose two categories of advice during the training stages–either that of an oracle who has knowledge of the exact scores of the N cadidates, or that of an expert who provides expertise based upon some policy function operating with limited knowledge. We ultimately find that policies learned off of expert advice were able to provide information over a wider range of the state space, enabling them to be much more successful than those learned off of the oracle in selecting the best candidate.

1 Introduction

In optimal stopping problems, an agent must determine the optimal time to choose a certain action to maximize a specified objective. Such situations are critical to understand for a number of human decision making problems [Robbins, 1970]. One such example, particularly relevant within the realms of dating and job-searching is the secretary problem. Though the explicit origins of the secretary problem are somewhat a mystery [Ferguson, 1989], it seems that a version of the problem was first introduced by Flood in 1949, before appearing in print form from potentially a separate source in Martin Gardner's 1960 Scientific American column Gardner [1960] (see also Gardner [1966]). Freeman [1983] point out alternative potential sources, like Gleason posing the problem in 1955, with Lindley providing the first published solution [Lindley, 1961]. We provide the formulation we use below (though a variety of extensions exist), before proceeding to discuss some existing literature surrounding the problem.

For completeness, here is the traditional secretary problem. An employer is trying to fill one vacant position. There are N candidates with scores u_i uniformly distributed between $[0, 1]$. The employer knows that the scores are uniformly distributed but does not know the scores of all the candidates in advance and does not know the range of the distribution. The employer can sequentially interview the candidates to have their individual scores revealed, but he must choose to instantly accept or reject and cannot change his mind afterwards. Ultimately, the employer receives a reward of 1 if he accepts the best candidate and a reward of 0 for every other outcome.

Preprint. Under review.

Ferguson [1989] provides a derivation for the optimal threshold-based strategy for an agent faced with this problem for a single round. In this threshold based approach, the employer rejects the first $r-1$ candidates and then accepts the first candidate whose score is larger than all of candidates present among the first $r-1$. For any choice of threshold r, the probability the candidate selected is the best one is given by: $P(r) = \frac{r-1}{n} \sum_{i=r}^{n} \frac{1}{i-1}$. For finite N, one can search for which value of r maximizes this quantity to determine the threshold. However, in the limit as n tends to ∞, the optimal threshold tends toward $\frac{n}{e}$.

A number of experimental studies have been done to investigate what humans actually do when faced with such a problem. Bearden et al. [2006] find that agents stop their search early and seem to overestimate the quality of applicants they see early on and give insufficient consideration to later applicants. Palley and Kremer [2014] similarly find that, particularly in cases where the subjects were given partial information, they tended to stop prematurely.

In more recent years, a number of studies have appeared within the realm of reinforcement learning applied to optimal-stopping problems [Kong et al., 2018, Goldstein et al., 2020]. Particularly, within the context of financial engineering, Fathan and Delage [2021] employ Double Deep Q Networks (DDQN) to determine optimal stopping policies in the case of option pricing and option exercise. Moreover, Goel et al. [2017] develop a sample-efficient way of trying to learn an optimal policy for certain classes of optimal-stopping problems. In a modified version of the secretary problem, Siboni attempts to train a reinforcement learning agent using DDQN and outperforms Monte Carlo simulations of thresholding-rule based performances. He particularly points out the benefit of learning the range of the distribution of candidate scores.

However, not much investigation has been done into the nature of the policy functions learned by these various methods and as to the structure of information they encode. An additional approach for many reinforcement learning problems is imitation learning [Abbeel and Ng, 2010, Ho and Ermon, 2016, Dixon et al., 2020, Le Mero et al., 2022], where an agent tries to learn an optimal policy based upon the actions of a teacher. As far as we can see, very little has been done within the realm of imitation learning for optimal stopping problems. We will use behavioral cloning [Pomerleau, 1988, Michie et al., 1990] framework for imitation learning in the context of optimal stopping problems.

In the traditional formulation of imitation learning, the teacher and the student has the same direct or indirect information about the state space. In practice though, that is rarely the case. In this paper, we propose studying which type of guidance is beneficial for an agent to learn an optimal policy function for the secretary problem. We can compare learning from the actions by experts, with different levels of knowledge,who act based-upon an optimized policy rule to learning from the actions of an all-knowing oracle who is aware of full collection of candidate scores in each round.

2 Environment and Model

Our state space $\mathcal{S} : [0,1] \times [0,1] \times [0,1]$ is defined by the time index of the candidate currently being interviewed: $\frac{i}{N}$ ranging from 0 to 1, the max score seen till date, and the new score of the candidate in the present round. We define an individual state $s \in \mathcal{S}$ to have components $s = (s_1, s_2, s_3)$. The actions taken from any state are either 0 for reject or 1 for accept, so $\mathcal{A} : \{0,1\}$. Given any state-action pair the reward the employer can receive is either 1 if he accepts the best candidate or 0 for every other outcome. That is, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \{0,1\}$. We provide the following transition rule for our function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. Note that if the employer accepts, the round terminates. Here, we will denote $\mathcal{P}(s_t, 1) = s_t$. If the employer rejects, $\mathcal{P}(s_t, 0) = (s_{1,t} + \frac{1}{N}, \max\{s_{2,t}, s_{3,t}\}, u_{new})$ where $u_{new} = u_{Ns_{1,t+1}}$ for the score of the candidate indexed at the next period of time. We define our policy function to be $\pi : \mathcal{S} \to P(\mathcal{A})$, where $\pi(1|s)$ corresponds to the probability of accepting the candidate in state s and $\pi(0|s) = 1 - \pi(1|s)$ corresponds to the probability of rejecting the candidate in state s.

3 Methods

We propose comparing the performance of three different policy functions. Namely, one policy function learned based upon oracle advice, and then two based upon experts following some sort of a policy: one based upon the optimal one-round thresholding strategy mentioned above and

then one based upon training an expert. We treat this study as if it was the extreme case where an agent was receiving 100 percent advice from either the oracle or expert(s). For the purposes of analysis, we discretize the state space. Since in our simulations we choose to have 20 candidates, we discretize our time space to have 20 points and then we choose to have 15 grid points for each of the score-based axes. For the policies that require training to learn, we choose to have $T = 600$ iterations. We choose to run $K = 5$ episodes per learning iteration to generate trajectories of the agent.

The oracle advice at any given state, based upon full knowledge of the score-distribution, is given as $o : \mathcal{S} \to \mathcal{A}$. Denoting the best candidate index out of 20 as b, $o(s) = I(\frac{b}{N} = s_1)$. Define $\mathcal{O}$ as the set of oracle-based trajectories. For the oracle-based policy function, we theoretically define the probability of acceptance at any given state as follows: $\pi_o(1|s) = \frac{\mathbb{E}_{\tau\sim\mathcal{O}}[\sum_{t=1}^{T} o(s_t(\tau))I(s_t(\tau)=s)]}{\mathbb{E}_{\tau\sim\mathcal{O}}[\sum_{t=1}^{T} I(s_t(\tau)=s)]}$ where τ represents one trajectory taken by the agent out of $\mathbb{O}$. Note that this is well defined in cases where the theoretical probability of reaching every state is nonzero, in practice, we choose to initialize the tally of number of times the state has been visited to 1.

Note that in the case of $N = 20$ agents, for the thresholding strategy discussed above, the optimal threshold is $r = 8$. We define $\pi_t(1|s) = I(s_1 \geq \frac{r}{N})I(s_3 > s_2)$.

Finally, we train an agent to learn an optimal policy using Double Deep Q Networks (DDQN), an approach that combines ideas from both Double Q Learning and Deep Q Learning [Van Hasselt et al., 2016]. We parameterize our Q function by a $2-$layer neural network with 64 neurons each. We choose a discount factor of $\gamma = 0.99$, though for our problem, use of a discount factor is somewhat unimportant. We can use gradient descent to update it via the following loss function: $L(\theta_i) = \mathbb{E}_(s, a, r, s')[\frac{1}{2}(r + \gamma Q(s', \arg\max_a Q(s', a; \theta_i); \theta_{i-1}) - Q(s, a; \theta_i))^2]$. One can compute the ultimate policy function from this Q value function by choosing the action that maximizes the Q value for a given state: $\pi_d(1|s) = \arg\max_a Q(s, a; \theta)$.

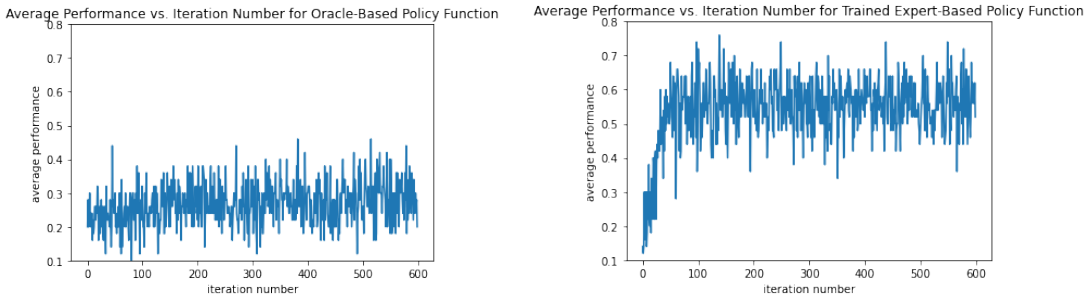
4 Results

We begin by observing that in the case of a random policy where an employer chooses a random index to stop at every round of the game, we would expect that he succeeds with probability $\frac{1}{N}$ in each round. Therefore, when simulating for $10,000$ rounds with 20 candidates, we expect that the random policy would generate around 500 successes and have an average performance of around 0.05.

Fig. 1a shows that the oracle seems to broadly learn something as the number of iterations increases and tends to perform than a random policy would. However, the performance does not grow near the stage of later iterations. We can contrast this with the expert trained by DDQN in Fig. 1b, where there is a much clearer sign of learning and broadly we already see signs of greater average performance compared with that of the oracle-based policy. We can now study the nature of the policy functions in comparison to each other. That is, of the oracle-based policy function in comparison to that of the experts' based upon the thresholding-based policy and the trained DDQN expert. We can see that in the case of the oracle-based policy in Fig. 2a, acceptance is only done with non-zero probability along the layer where the new score of the candidate present in the round is close to 1. This makes sense given that that is the only region of the state space where the oracle indicates to the agent that he should accept that candidate. We can also observe that the region of the state space that ends up getting having non-zero probability of acceptance is mainly in the space where the max score till date is larger as time increases and we do not end up seeing as many cases of acceptance where early on in time there is a larger value of max-score till date where the candidate also ends up seeing a large current score value of close to 1.

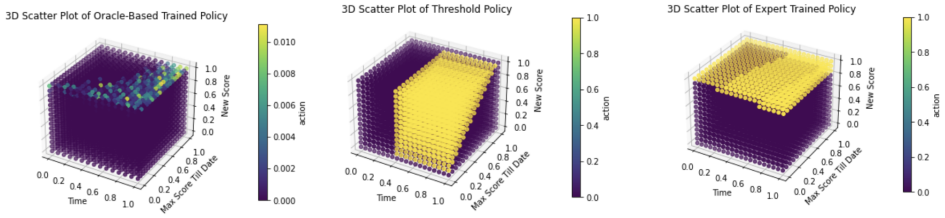
We can contrast this policy function with that of the thresholding-based one in Fig. 2b which was optimal for the one round version of the secretary problem. Here we can see that the agent simply gathers information on the max score of the first r candidates and then only accepts the next candidate of someone with a score larger than the max score till date.

Lastly, we can study the policy learned by the expert trained through DDQN in Fig. 2c. Here we can see clear signs of distribution-based learning as early on, the agent learns to accept scores where the



(a) (b)

Figure 1: Average performance from 50 simulations per learning iteration. (a) Results based upon the policy function learned from oracle information. Note, that when simulating to test average performance, one is not using the oracle within those 50 rounds of simulation itself, but rather is applying the policy function learned up to that stage. (b) Results based upon the policy function learned from the trained expert using DDQN. Note the performance improvement in the latter case.



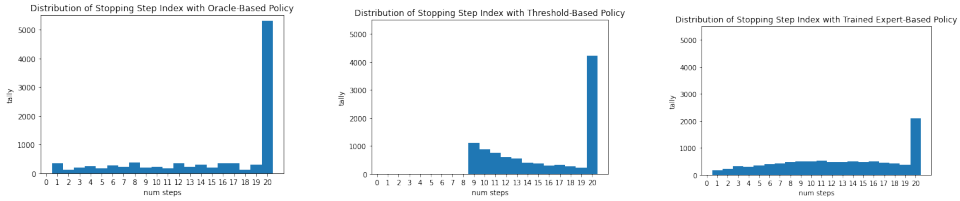
(a) (b) (c)

Figure 2: Note that the heat map corresponds to the probability of selection at that state within an episode. The three panels correspond to the policy functions trained by actions of (a) the oracle, (b) threshold strategy expert and (c) the DDQN expert, respectively.

current value is close to 1, which it would not have known to do, if it did not have knowledge of the relative distribution of scores. Moreover, we can see that as time goes on, the agent slightly lowers it threshold of acceptance, which is interesting given the $0-1$ scoring nature of our problem.

Ultimately, we can study the performance of each of these policy functions over $10,000$ simulations by looking at Table 1 It is unsurprising that the DDQN-based expert policy ultimately ends up performing the best given that it ends up learning about the distribution and thus is able to advise the agent to accept early on when it sees high scores unlike the thresholding-based policy which does not assume knowledge of distribution and must use the first r interviews to learn about the nature of the distribution based on relative ranks and scores of the candidates it sees. It also makes sense that both of those expert-based policies end up performing better than the oracle-based policy where the agent is only advised to accept on a thin slice of the state space where the scored presented in front of the candidate is high. This policy ends up performing better than random given that it recommends the candidate only to accept large values of scores with some probability, and nothing else (and thus also provides information about the nature of the distribution). However, even within this region of the state space where the current score value is high, the probabilities with which the agent ends up accepting at that given point in state space are still quite low (given that upon tallying, the number of times the best candidate is located at that exact point in the state space is quite small). We also provide the distribution of the candidate index at which the agent chooses to stop in Figs. 3a, 3b, and 3c. We can see that in the thresholding case, the agent only starts stopping after index r, while in the others, the agent stops at almost all the indices, but for all three policies, we see that in a large number of cases, the agent only stops when forced to by reaching the last candidate, which means the agent tends to wait till the very end and rejecting all options provided earlier.

It is important to note that the oracle-based policy function is somewhat sensitive to how one chooses initialization for states that have not been explored. Nonetheless, for a variety of initializations we tried, the oracle-based policy function still performed worse than those of the threshold-based and



(a) (b) (c)

Figure 3: Distribution of stopping step over $10,000$ simulations. The three panels correspond to the policy functions trained by actions of (a) the oracle, (b) threshold strategy expert and (c) the DDQN expert, respectively. Note that the fraction of cases where the agent does not stop early reduces from left to right.

Teacher	Success rate	Fraction stopping at N
No teacher, random policy	0.0500	0.0500
Oracle	0.2873 ± 0.0089	0.5299 ± 0.0098
Threshold-based expert	0.3841 ± 0.0095	0.4208 ± 0.0097
DDQN-based expert	0.5629 ± 0.0097	0.2098 ± 0.0080

Table 1: Summary of results, comparing different policies with $10,000$ simulations in each case. The errors correspond to 95% confidence interval.

DDQN-based policy. Of course, the oracle-based policy is also sensitive to how we specify the nature of the update rule–we chose the simple-minded tally-based approach for the sake of preliminary exploration. Moreover, it is worth noting that perhaps with an extremely large number of iterations and finer-discretization, the oracle would end up recommending acceptance anywhere it saw an extremely large score presented in front of the agent with a large enough probability, but for the sake of comparison we chose to train each of these policy functions for the same (small) number of iterations. Also, in the formulation of the secretary problem we chose, the agent only receives a score of 1 in the case where it accepts the absolute best candidate, and there is not gradation provided for accepting candidates with decently high, but not globally maximal scores. This ends up creating a very sparse set of information available to an agent trying to learn an optimal policy. We would conjecture that with a scoring mechanism where an agent receives the score of the candidate that it accepts, the performance of the oracle and DDQN trained agents would be larger, with the expert-based one performing better.

5 Conclusion

Ultimately, we found that the expert that provided guidance based upon the DDQN-learned policy ultimately ended up performing the best relative to the other threshold-based policy and the oracle-based policy. This was a result of the fact that the DDQN expert was able to utilize distribution-based information early on, unlike the threshold-based expert whose strategy is optimal for an agent only playing this game one round.

The oracle-based policy ends up also providing some distribution-based information to the agent and thus ends up performing better than random, but nonetheless does not provide a structured way for the agent to learn what the optimal course of behavior in a future round without any assistance should be. The mismatch between the state/observation space of the oracle and the agent produces this loss performance, despite the superior knowledge of the oracle.

6 Discussion and Extensions

It would be interesting to test whether these findings hold for the reinforcement learning agent in cases where it is not so simple to learn other information about the nature of the distribution. More importantly, general investigations of cases where the teacher and the student see the world

differently Cai et al. [2021] will shed light on under what conditions feedback can be properly incorporated into the student's policy function.

In the broader context of the interaction between reinforcement learning and economic sciences, a much greater emphasis has been placed on data-driven policies [Mosavi et al., 2020, Charpentier et al., 2021]. We hope that reformulation of the problems in terms of learning dynamics of agents also sheds some light on the efficacy of different strategies employed in our society.

Acknowledgement

SS would like to thank Albert Chiu for his feedback in the early stage of this this project.

References

Pieter Abbeel and Andrew Y Ng. Inverse reinforcement learning., 2010.

J Neil Bearden, Amnon Rapoport, and Ryan O Murphy. Sequential observation and selection with rank-dependent payoffs: An experimental study. *Management Science*, 52(9):1437–1449, 2006.

Xin-Qiang Cai, Yao-Xiang Ding, Zi-Xuan Chen, Yuan Jiang, Masashi Sugiyama, and Zhi-Hua Zhou. Seeing differently, acting similarly: Imitation learning with heterogeneous observations. *arXiv preprint arXiv:2106.09256*, 2021.

Arthur Charpentier, Romuald Elie, and Carl Remlinger. Reinforcement learning in economics and finance. *Computational Economics*, pages 1–38, 2021.

Matthew F Dixon, Igor Halperin, and Paul Bilokon. Inverse reinforcement learning and imitation learning. In *Machine Learning in Finance*, pages 419–517. Springer, 2020.

Abderrahim Fathan and Erick Delage. Deep reinforcement learning for optimal stopping with application in financial engineering. *arXiv preprint arXiv:2105.08877*, 2021.

Thomas S Ferguson. Who solved the secretary problem? *Statistical science*, 4(3):282–289, 1989.

PR Freeman. The secretary problem and its extensions: A review. *International Statistical Review/Revue Internationale de Statistique*, pages 189–206, 1983.

Martin Gardner. A 5th collection of brain-teasers. *Scientific American*, 202(2):150, 1960.

Martin Gardner. *New mathematical diversions from Scientific American*. Simon and Schuster, 1966.

Karan Goel, Christoph Dann, and Emma Brunskill. Sample efficient policy search for optimal stopping domains. *arXiv preprint arXiv:1702.06238*, 2017.

Daniel G Goldstein, R Preston McAfee, Siddharth Suri, and James R Wright. Learning when to stop searching. *Management Science*, 66(3):1375–1394, 2020.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Weiwei Kong, Christopher Liaw, Aranyak Mehta, and D Sivakumar. A new dog learns old tricks: Rl finds classic optimization algorithms. In *International Conference on Learning Representations*, 2018.

Luc Le Mero, Dewei Yi, Mehrdad Dianati, and Alexandros Mouzakitis. A survey on imitation learning techniques for end-to-end autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

Denis V Lindley. Dynamic programming and decision theory. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 10(1):39–51, 1961.

Donald Michie, Michael Bain, and J Hayes-Miches. Cognitive models from subcognitive skills. *IEE control engineering series*, 44:71–99, 1990.

Amirhosein Mosavi, Yaser Faghan, Pedram Ghamisi, Puhong Duan, Sina Faizollahzadeh Ardabili, Ely Salwana, and Shahab S Band. Comprehensive review of deep reinforcement learning methods and applications in economics. *Mathematics*, 8(10):1640, 2020.

Asa B Palley and Mirko Kremer. Sequential search and learning from rank feedback: Theory and experimental evidence. *Management Science*, 60(10):2525–2542, 2014.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Herbert Robbins. Optimal stopping. *The American Mathematical Monthly*, 77(4):333–343, 1970.

Nima Siboni. `https://medium.com/deepmetis/deep-reinforcement-learning-for-optimal-stopping-problems`

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.