Getting Good at Googol: Learning Optimal Stopping Theory

Erica Detemmerman, Erica Dominic, Isaac Haberman
ed1762@nyu.edu, ehd255@nyu.edu, ijh216@nyu.edu
New York University

Abstract

The game of googol is the quintessential optimal stopping problem: the player's only task is to determine the best time to quit the game. We trained four agents to play the game of googol using common reinforcement learning algorithms: first visit Monte Carlo, SARSA, Q-learning, and deep Q-learning. All agents learn to play the game of googol with success rates between 19% and 37%, with the deep Q-learning agent proving the most successful. Furthermore, we demonstrate that all agents are able to transfer the knowledge accumulated while playing one version of the game to another version of the game. We also conducted a small number of trials in which human participants were asked to play the game of googol. Collectively, human participants attained a success rate of 35%, comparable to the deep Q-learning agent; however, human players' strategies do not correspond with strategies observed in the reinforcement learning agents, as evidenced by trends in stopping choices.

I. Introduction

In the late 1950s and early 1960s, a popular puzzle circulated around the mathematical community. The puzzle was simple to state, but had a striking solution. In February of 1960, Martin Gardner published one version of the puzzle in his *Scientific American* column "Mathematical Games." He called it "the game of googol" [3].

In this paper we will use the following version of the game: a sequence of n numbers is hidden from the player. Each number is an integer between one and googol (1^{100}). The numbers are revealed to the player one at a time, and with each reveal the player chooses whether to continue viewing the sequence or stop on the current number. The player wins if they choose to stop on the highest value in the sequence. Importantly, the player does not know which values will appear later in the sequence and cannot revert to a previously seen value.

A Mathematical Approach

Given these rules, it is possible to mathematically derive a stopping rule that maximizes a player's probability of success. The optimal policy entails passively viewing the first x numbers in the sequence while tracking the maximum value M. The player then views the remainder of the sequence and stops on the first value that is greater than M. The probability of winning the game with this policy is

$$x \int_x^1 \left(\frac{1}{t}\right) dt = -x \ln x$$

The optimal $x = 1/e \approx 0.367879...$, meaning the player should passively observe the first 37% of the sequence. The probability of success using this strategy is also $1/e \approx 0.367879....$ Remarkably, the 37% success rate holds regardless of sequence length [2].

Related Research in Cognitive Science

The game of googol is one example of an optimal stopping problem: a player must choose when to take a particular action in order to maximize an expected reward. Due to its mathematical formulation, the game of googol lends itself to probabilistic and statistical analysis, yet optimal stopping problems extend far beyond the realm of mathematics.

Humans frequently encounter various incarnations of the optimal stopping problem: when interviewing a series of candidates to fill a position, when apartment hunting in a fast-paced city, or when scrolling through YouTube deciding which video to watch.

Many factors influence human behavior in situations involving an optimal stopping problem: familiarity with the environment, how quickly the environment is changing, and the relative value of exploiting known outcomes versus the cost of exploring new possibilities [1].

Humans frequently face complex situations with limited knowledge of an environment that changes over time. It has been shown that humans dynamically update their estimates of rewards associated with chosen actions, tending to explore more when there is more time allowed for a task or more total items to sample [4].

Exploring is also influenced by factors such as minimizing cognitive load, desire to obtain an outcome more quickly, or negative experiences. For example, if someone experiences a negative experience early on, this may lead to risk-aversion, lower knowledge of the environment, and undersampling, a phenomena that has been called the "hot stove effect" [5].

Overview

In order to further explore optimal stopping theory and the learning of stopping rules, we implemented five computational models that play the game of googol. Section II describes the models, as well as the methods used for training and evaluation. Section III describes the trials we conducted in which human participants played the game of googol. Section IV discusses the results from model evaluations and human trials; we analyze the strategy of each model and compare models' strategies to trends seen in human data. Finally, section V summarizes our findings and suggests directions for future work.

II. Models and Methods

For this project, we implemented the game of googol as well as five agents. The source code, including tuned hyperparameter settings, and can be found in the repository `https://github.com/isaachaberman/DeepGoogol`.

The Agents

Rule-Based Agent. The rule-based agent follows the 37% rule described in section I. The agent passively observes the first 37% of the sequence while tracking the maximum value M. The agent then views the remainder of the sequence and chooses to stop on the first value that is greater than M. If no value greater than M exists in the remainder of the sequence, the agent views the entire sequence and by default stops on the final value.

Monte Carlo Agent. The Monte Carlo agent uses the first visit Monte Carlo algorithm. The agent plays several episodes to their conclusion, notes the result, then attributes the outcome to all game states encountered during that episode. Return is calculated with a discount factor $\gamma = 0.9$. To encourage exploration, the agent uses an epsilon-greedy strategy. Initially $\epsilon = 0.1$ and this value decays over time.

SARSA Agent. The SARSA agent uses the general paradigm of temporal difference learning in conjunction with the SARSA update rule. The update rule computes Q values for each state-action pair, then takes the action that maximizes expected reward. The SARSA agent uses a learning rate of $\alpha = 0.01$, a discount factor of $\gamma = 0.9$, and an epsilon greedy strategy where initially $\epsilon = 0.1$. Both the learning rate and epsilon value decay over the course of training.

Q-Learning Agent. The Q-learning agent resembles the SARSA agent, but uses the Q-learning update rule to compute Q values for each state-action pair. For the purposes of comparability, the Q-learning agent uses the same α, γ, ϵ, and decay rates as the SARSA agent.

Deep Q-Learning Agent. The deep Q-learning agent uses deep neural networks that generate actions from an input state space. Two networks are involved: a policy network and a target network. Each network is three layers, including a hidden layer with 32 units. During training, the algorithm draws a random sample from a buffer of 2,500 games it has previously played. The networks used a batch size of 256, Huber loss, and the RMSprop optimizer [6]. The deep Q-learning agent also used the discount and exploration parameters $\gamma = 0.9$ and $\epsilon = 0.1$.

A note on representing the state of the game: technically a game state includes the entire sequence of numbers the agent has encountered up until a given point in the game. Given a long sequence length and a large range of possible numbers, the state space of the game of googol is very large. Even when an agent plays several thousand games, the agent is unlikely to encounter the exact game state (i.e. the exact sequence of cards in the same order) multiple times.

To facilitate training in such a large state space, each agent uses a function to summarize the game state. We tested several such functions that incorporate different combinations of game state attributes, e.g. how many cards have been viewed, the current value, and the maximum value encountered thus far. The functions also bin these values; this effectively groups similar game states together and reduces the size of the state space.

Training

All trainable agents were trained on games using sequences of length 50 with numbers in the range 1 to $100,000$. Sequence values were distinct and drawn from a uniform distribution. The game of googol has a very stringent victory condition: the player must stop on the highest card, and no reward is given for stopping on the second highest card, etc. To facilitate learning during this training phase, agents were rewarded for stopping on any of the three highest values in the sequence. All four trainable agents were trained on $500,000$ games.

Transfer Learning

After the initial training phase, each agent was introduced to a new version of the game. See table 1 for a description of the different game versions. Note that the uniform and normal distributions were discretized. "Uniform" implies that all elements in the range were equally likely to appear. "Normal" indicates that numbers were sampled from $\mathcal{N}\left(50000, 17000^2\right)$, rounded to the nearest integer, and discarded if the result was outside the game's specified range.

The agent retained its configuration from the initial training phase, then trained on $10,000$ episodes of game i where $0 \leq i \leq 7$. The extra training gave the agent limited experience with the new version of the game before entering the evaluation phase, and allowed us to investigate whether the agent was able to transfer knowledge from one optimal stopping problem to another.

Evaluation

After initial training on $500,000$ episodes of the original game and additional training on $10,000$ episodes of the altered game, each agent was evaluated on $10,000$ trials of the altered game. During the evaluation phase, the agent was rewarded only if it chose to stop on the highest number in the sequence.

Over the course of evaluation, we tracked the success rate of each agent; these results are displayed in table 2. We also recorded where in the sequence the agents chose to stop; these results are shown in individual bar graphs in section IV.

III. Human Trials

In addition to the five agents, a small group of human participants were asked to play five to ten rounds of the game of googol. In the human trials, players won the game only if they stopped on the highest value in the sequence. In order to make the game more tractable for a human player, the sequence length was reduced to 20. Values still ranged from 1 and $100,000$.

Human players were informed of the sequence length, the range of the numbers, and the goal of the game. Participants were not given any information on the distribution of the values or optimal stopping theory.

Sequence values were written down or displayed on a monitor one at a time. Once a player chose to move past a number, it was removed from view. At the end of each game, the player was told only whether they won or lost.

After each game, we recorded the outcome of that round and where in the sequence the player chose to stop.

IV. Results and Discussion

For each agent, rates of success on the different game versions are displayed in table 2. Each agent's stopping choices over the course of $10,000$ evaluations on Game 0 are shown in the bar graphs below.

Rule-Based Agent

The rule-based agent required no training, and was evaluated on $10,000$ episodes of Game 0. The agent achieved a 37% success rate, a result consistent with the mathematical argument outlined in section I.

Figure 1 illustrates where in the sequence the rule-based agent chose to stop over the course of $10,000$ evaluations on Game 0. Because the highest value is equally likely to appear anywhere in the sequence, the maximum value appears in the first 37% of the sequence approximately 37% of the time. In those cases, the agent views the entire sequence and never encounters a better value than the maximum found in the first 37%. This accounts for the spike in the bar graph at stopping index 50.

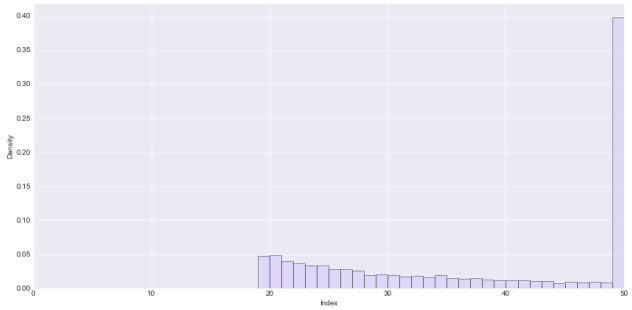

Figure 1: Stopping choices for the rule-based agent

As seen in table 2, the success rate of the rule-based agent matched or exceeded the success rate of all other agents, and required no training. However, the rule-based agent did require a mathematical proof and manual coding of the optimal strategy. Furthermore, the optimality argument no longer holds when certain features of the game are altered, for instance the if the numbers are generated with replacement, or from a non-uniform distribution. The following sections explore results from the four trainable agents, none of which required explicit implementation of a predesigned strategy.

	Range (Inclusive)	Sequence Length	With Replacement?	Distribution	Rewarded for k Highest
Training	1 – 100,000	50	No	uniform	k = 3
Game 0	1 – 100,000	50	No	uniform	k = 1
Game 1	1 – 1,000	50	No	uniform	k = 1
Game 2	1 – 10,000	50	No	uniform	k = 1
Game 3	1 – 1,000,000	50	No	uniform	k = 1
Game 4	1 – 100,000	25	No	uniform	k = 1
Game 5	1 – 100,000	100	No	uniform	k = 1
Game 6	1 – 100,000	50	Yes	uniform	k = 1
Game 7	1 – 100,000	50	No	normal	k = 1

Table 1: Versions of the googol game used to demonstrate transfer learning; gray shading indicates how the game differs from the training game

Agent	Game 0	Game 1	Game 2	Game 3	Game 4	Game 5	Game 6	Game 7
Monte Carlo	19%	19%	19%	19%	36%	11%	20%	45%
SARSA	23%	22%	23%	22%	41%	13%	23%	47%
Q-learning	23%	24%	24%	24%	42%	13%	24%	46%
Deep Q-learning	37%	37%	36%	37%	57%	18%	39%	43%

Table 2: Success rate out of 10,000 evaluation games of the specified version

Monte Carlo Agent

The Monte Carlo agent attained a success rate of only 19% on Game 0. Though below optimal, the agent performed better than random, which would yield a success rate of $1/50 = 2\%$. Moreover, the agent was able to sustain the 19% success rate when the range of numbers varied (in Games 1, 2 and 3) and showed improvement when sequence values were generated from a normal distribution (Game 7). Due to the large range of values and comparatively small sequence length, allowing duplicate values had little effect on success rate (Game 6).

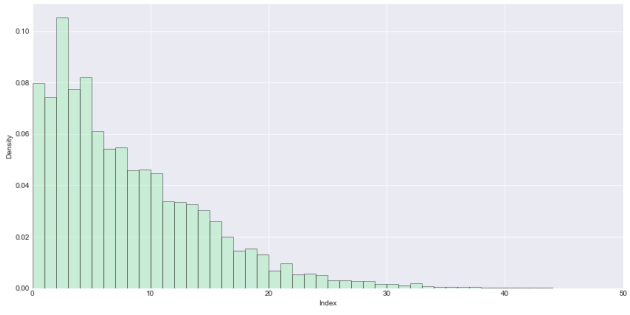

Figure 2: Stopping choices for the Monte Carlo agent

Figure 2 shows the distribution of choices the Monte Carlo agent made during 10,000 evaluations on game 0. Unlike the rule-based agent, the Monte Carlo agent strongly favors early stopping indices. The Monte Carlo agent entirely avoids stopping on the last few values, a behavior unique to this agent.

SARSA Agent

The SARSA agent attained a success rate of 23%, slightly higher than that of the Monte Carlo agent. Like the Monte Carlo agent, the SARSA agent was able to sustain or improve that success rate when the sequence length or distribution was varied.

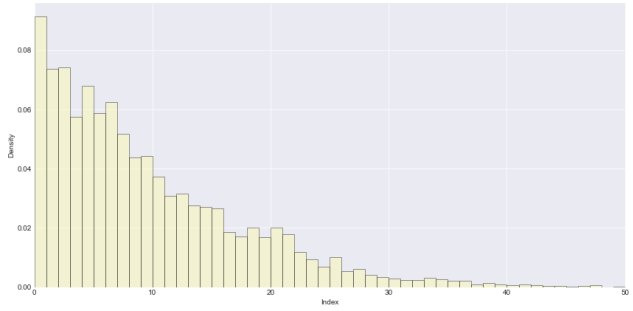

Figure 3: Stopping choices for the SARSA agent

The SARSA and Monte Carlo agents exhibit similar behavior. Both choose later stopping indices with decreasing probability, and no specific indices are significantly preferred.

Q-Learning Agent

The Q-learning agent attained roughly the same success rates as the SARSA agent. This was expected, due to the similarity of the algorithms in the temporal difference framework. Like the SARSA agent, the Q-learning agent was able to transfer knowledge between various versions of the game.

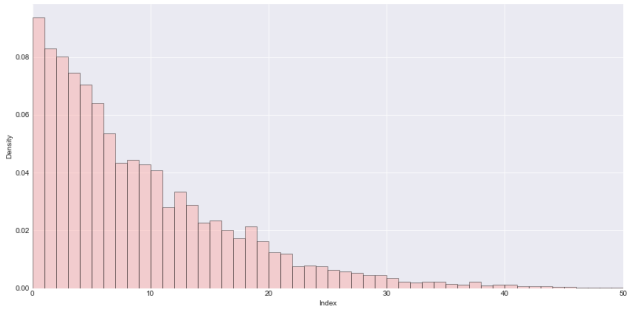

Figure 4: Stopping choices for the Q-learning agent

The Q-learning agent behaves similarly to the SARSA agent in terms of stopping choices. The agent tends to stop on values that appear earlier in the sequence, and does not favor earlier indices as strongly as the Monte Carlo agent.

Deep Q-Learning Agent

The deep Q-learning agent was the only trainable agent that attained a 37% success rate through reinforcement learning. The deep Q-learning agent was also able to transfer knowledge and maintain a high success rate among the different versions of the game. With the exception of game 7, which used a sequence generated by a normal distribution, the deep Q-learning agent achieved the highest accuracy across all the evaluation games. In most cases, the agent matched or outperformed the rule-based agent.

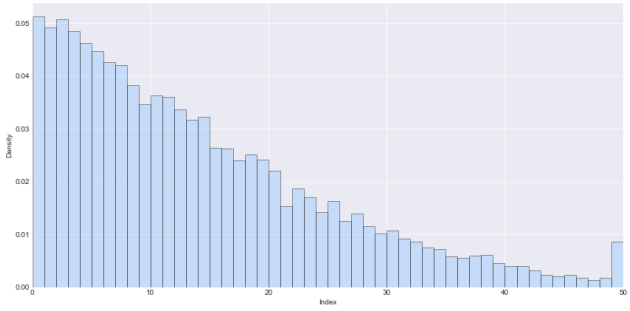

Figure 5: Stopping choices for the deep Q-learning agent

While the rule-based agent and deep Q-learning agent achieved the same success rate on Game 0, the two agents have very different strategies, as evidenced by their stopping choices.

The stopping choices for the deep Q-learning agent follow the same trend as the Monte Carlo, SARSA, and Q-learning agents. However, the deep Q-learning agent is slightly more "patient" compared to the other trainable agents, meaning the deep Q-learning agent is more prone to choosing numbers that appear later in the sequence. Unlike the other trainable agents, it is evident that the deep Q-learning agent occasionally views the entire sequence, having missed the maximum earlier in the sequence. This leads to a spike at stopping index 50 which is reminiscent of the rule-based agent's behavior. This "patience" could account for the deep Q-learning agent's 37% success rate.

Human Trials

Finally, in figure 6 we display the results of the human trials. Eight humans played a total of 48 games (version 0, as described in table 1). Human players collectively won 17 of the 48 games, giving the human players a combined success rate of approximately 35%. This result on par with the rule-based agent and the deep Q-learning agent.

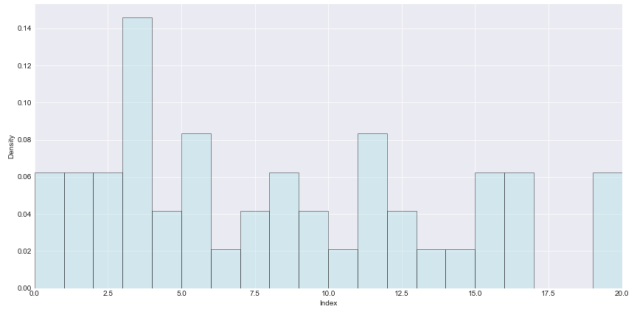

Figure 6: Stopping choices made by human participants

The strategies used by human players do not strongly resemble the strategies used by any of the computational models. This may indicate that human players and reinforcement learning algorithms have very different approaches to optimal stopping problems. Human players may also be effected by previous experience with stopping problems, or be unwilling to expend cognitive resources on the problem and instead settle for a non-optimal "good enough" solution to the problem.

It is also worth noting that a relatively small amount of data was collected from human participants.

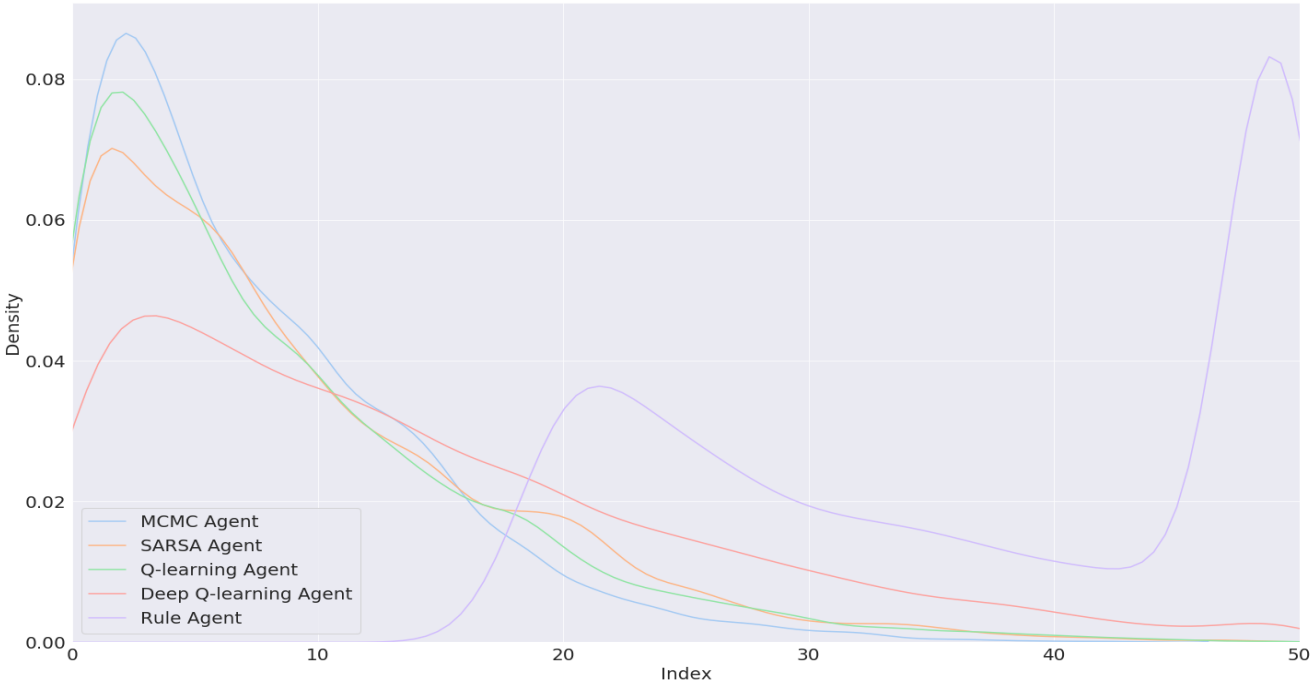

Figure 7: Density plot of stopping choices for the five computational models

V. Conclusion

Summary

Of the four reinforcement learning agents, the deep Q-learning agent attained the highest success rates across the different versions of the game. The deep Q-learning agent uses a deep neural network to approximate Q-values instead of tracking individual state-action pairs explicitly, and is thus well-suited to handling games with large state spaces. This may explain the deep Q-learning agent's relative success.

The other three reinforcement learning agents did not attain a 37% success rate, even when trained to convergence. This may be due to the function used to summarize the game state (discussed in section II). It may also be due to the fact that initial training rewarded the agent for stopping on any of the three highest values. Though this helped facilitate training, the agent suffered a drop in success rate when it transitioned from the training game to game 0, which only rewarded the agent for stopping on the single highest value.

Although some of the reinforcement learning algorithms failed to reach the optimal success rate, they each have a flexibility which the rule-based agent lacks. The reinforcement learning algorithms do not require a mathematical proof, nor do they require a hard coded strategy to be manually implemented. Furthermore, the mathematical argument supporting the 37% strategy no longer holds if the numbers in the sequence are generated with replacement, or from a non-uniform distribution. The results in table 2 demonstrate that reinforcement learning agents are capable of transferring knowledge from one version of the game to another, even when the argument for the 37% rule is invalidated.

Figure 7 displays the trends in stopping choices for the five computational models. (As humans played a version of the game with shorter sequences, those results were not included in the combined density plot.) The plot clearly shows the relative "patience" of the deep Q-learning agent compared to the other agents using reinforcement learning.

Future Work

To better compare results from human trials and computational models, it is necessary to collect more data from human participants. With more trials, trends in human players' stopping choices may be become apparent. If human player strategies still differ from the approaches used by the computational models, it would be interesting to investigate the effect of search cost on computational models. Incorporating search cost may allow the computational models to simulate a human's tendency to conserve cognitive resources and undersample.

References

[1] Jonathan D. Cohen, Samuel M. Mcclure, and Angela J. Yu. Should I stay or should I go? How the human brain manages the trade-off between exploitation and exploration. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1481):933–942, 2007.

[2] Thomas S. Ferguson. Who solved the secretary problem? *Statistical Science*, 4(3):2892–296, 1989.

[3] Alexander V. Gnedin. A solution to the game of googol. *Ann. Probab.*, 22(3):1588–1595, 07 1994.

[4] M.Z. Juni, T.M. Gureckis, and L.T. Maloney. Don't stop 'til you get enough: Adaptive information sampling in a visuomotor estimation task. In Carlson L., Holscher C., and Shipley T., editors, *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*, pages 2854–2859. Cognitive Science Society, 2011.

[5] A.S. Rich and T.M. Gureckis. The value of approaching bad things. In P. Bello, Guarini M., McShane M., and Scassellati B., editors, *Proceedings of the 36th Annual Conference of the Cognitive Science Society*. Cognitive Science Society, 2014.

[6] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.