HIGH SCHOOL EDITION
Journal of Student Research
Volume 12 Issue 3 (2023)

Offline Model-Learning by Learning Legal Moves in Othello Through Synthetic Negative Reinforcement

Johnny Liu[1] and Ganesh Mani[#]

[1]Thomas Jefferson High School for Science and Technology, USA
[#]Advisor

ABSTRACT

One of the biggest dependencies of reinforcement learning is sample efficiency since reinforcement learning agents tend to use an excessively large number of episodes to train. These episodes can be expensive, especially when they are conducted in an online manner. Model-Based Reinforcement Learning (MBRL) and offline learning are two methods to aid this problem; MBRL has been shown to improve sample efficiency while offline learning can reduce the number of online episodes needed by substituting them with less expensive offline ones. However, the use of these two methods together is more challenging, encountering issues such as training off of incomplete distributions. We explore these challenges by testing different combinations of offline and online model-learning through learning the legal moves in the board game Othello. During this process, we encounter the additional challenge of only positive reinforcement where offline episodes only provide information about legal moves. To address this problem, we propose a method named synthetic negative reinforcement which uses pre-existing agent knowledge to make up for the lack of information on illegal moves. Our results demonstrate the efficacy of offline learning using synthetic negative reinforcement on robust distributions of offline data, with agents achieving greater than 97% accuracy predicting the legality of moves. We also demonstrate the evident obstacle that skewed distributions provide to offline model-learning.

Introduction

Sampling efficiency has long been a problem in Reinforcement Learning (RL). While humans and animals may only need a couple of tries to learn a simple task, reinforcement learning is known to need an incredibly high number of trials (LeCun, 2022). This comparison with humans is made because RL has long been "influential in characterizing human learning" (He et al., 2022). The high number of trials needed especially becomes a problem in traditional online RL where the agent has to interact with the environment to learn. This interaction might be very expensive, which makes sampling efficiency even more critical (Wang et al., 2023).

One approach to increase sampling efficiency is to use Model-Based Reinforcement Learning (MBRL) which is where a model of the environment is constructed and learning is then conducted on the model (Sutton & Barto, 2018). Since the model can replace the need for real interaction with the environment if it is sufficiently accurate, having efficient model learning can lead to higher sampling efficiency. In previous studies, this has been shown (Kidambi et al., 2020).

Another approach is to avoid new interaction with the environment entirely, which is called offline RL. This is done when the agent learns by "watching" episodes which can come from pre-existing data (Sutton & Barto, 2018; Levine et al., 2020). Intuitively, this approach makes sense as humans might watch some examples before attempting something by themselves to largely increase their chances of success. However, according to Kidambi et al. (2020), using offline learning with model-based methods poses additional challenges. One challenge they identified is the distribution issue, where the model constructed from a limited dataset might be skewed leading to difficulties for the agent. Recently, the sentiment for offline RL has shifted more to using

offline learning as a starting point instead of completely replacing online RL (Agarwal et al., 2022). We are curious the extent to which offline model-learning is effective in supplementing online model-learning with regards to sample efficiency and different offline distributions.

While the sampling efficiency problem primarily exists in complex real-world applications, due to the lack of computing resources, the environment chosen to test offline model-learning in this study is the board game Othello. This choice was made because Othello boasts a large state-space of 10^{28} reachable possibilities, which is magnitudes greater than some other popular environments like Tic Tac Toe while not having such incredibly complex dynamics like some real-world applications to the point where training is too computationally intensive (Allis, 1994).

For this study, we will be conducting model-based learning through learning a crucial part of the state-transition function of Othello, the legal moves. The state-transition function in MBRL is a function that represents the probabilities of there being a transition from one state to another (Kim et al., 2022). Besides the legal moves, the other aspect of the state-transition function is the effect of the move on the environment, which is ignored due to the much more expensive state-transition function model needed to represent that.

In addition to the distribution problem, another challenge specific to conducting model-learning is identified through offline learning in Othello, the existence of Only Positive Reinforcement (OPR). OPR is the problem where sample games that the agent learns from provide information about the legal moves but nothing about the illegal moves. In the online case, the agent would get feedback on illegal moves through interacting with the environment, but this is not possible as only legal moves are played in sample games. As humans, this does not seem like so much of a problem, since we recognize patterns and thus can discern which moves are illegal as well as legal. However, as illustrated later, this problem is much more difficult for machines. In this study, we present a method of addressing the challenge of OPR through a method of creating synthetic negative reinforcement based on the agent's own prediction.

In summary, the contributions of this study consist of the following. First, we analyze the efficacy of using different amounts of offline model-learning from different distributions to supplement online model-learning. Second, we present a novel way to deal with the problem of Only Positive Reinforcement, which occurs in offline learning, through creating synthetic negative reinforcement based on the agent's previous experience.

Related Work

Othello has long been an application of RL. In 2013, van der Ree & Wiering showed how pure reinforcement learning algorithms, such as SARSA and Q-learning, could outperform benchmark players such as the random algorithm and the weighted squares algorithm. New strong programs have also been developed with Monte Carlo tree search methods, such as UCThello, which have been able to put up a fight against state-of-the-art heuristic search programs (Merkel, 2023; Buro, 2003).

These programs are all not model-based and directly encode the dynamics of the environment, such as legal move finding, straight into the engine. This makes sense as there is no reason to model-learn the rules of the game if the goal is to solely maximize playing strength. This approach is done here to study offline model-learning through the suitable environment dynamics of Othello and better model a standard human learning process where there is no intrinsic knowledge.

With regards to offline model-learning, there has been some work done outside of Othello. MOReL learns a pessimistic model to combat the distribution problem (Kidambi et al., 2020). Kim et al. (2022) proposed a technique of fitting the state-transition function's derivative. Both are presented as effective methods to either increase accuracy or increase sample efficiency.

To the best of our knowledge, there has been no direct solution to OPR. According to online articles and forums, the method of dealing with this problem is to label or mask illegal moves with pre-existing

knowledge so they are not picked by the agent (Xiang, 2021; István, 2017). This is counterintuitive to the model-learning problem, since labeling illegal moves requires external knowledge, whereas the online and offline methods in this study do not use any of this external information.

Methods

Data

Data in this study comes in two forms. In online learning, the agent interacts with a simulated environment. This is possible as the rules of Othello are completely known and a script can be written to perfectly simulate the environment. This script encodes the rules of Othello which consist of the following. The two players, black and white, start on a default starting board with 2 white tokens and 2 black tokens. Black plays first. During a turn, a player must play a legal move if it exists. If the player does not have a legal move, their turn is skipped. A position is a legal move if there is a sequence of tokens horizontally, vertically, or diagonally that contains at least one of the opponent's tokens and ends with the player's own token. Playing a move there will flip all of the opponent's tokens in this sequence to the player's color. Turns alternate until both players don't have legal moves. The player with the most tokens of their color on the board at the end of the game wins. All scripts used can be found in the GitHub at the end of the methods section.

In offline learning, the agent uses sample games (also called episodes) to train. These episodes come from two sources.

1. Self-generated games consisting of random distribution games (games played absolutely randomly, no preference of moves), and low-index skewed distribution games (moves with low index are preferred).
2. Human expert matches from the publicly available database from the French Federation of Othello (FFO). Since these high-level games come from the Wthor (.wtb) format, we pre-process them with a script based on the documentation from the FFO (Quin & Nicolet; Fédération Française d'Othello, 2023).

The random games should be an accurate representation of the overall distribution given enough episodes, while games played with low-index move preference are a skewed portion of the overall distribution. The expert human games should also be a skewed portion of the overall distribution as they are played at a high-level with the objective to win.

Agent

The agent in this problem will be attempting to learn an essential aspect of the Othello environment, namely the legal moves. To do this, the agent must first be able to "sense" the environment. We define the board as two 8x8 matrices where each square in each matrix is indexed as an integer from 0 to 63. The first matrix is the current player's color and the second matrix contains the opponent's color. Each square in a matrix contains 1 if there is a token of that color at that board position or 0 if not. Each pair of these two matrices represents a state. Each integer from 0 to 63 represents a position, or possible move. In this way, the agent can "sense" the environment by reading in these states and actions.

Second, the agent must be able to store and remember what it learned. To do this, the agent has its own state-transition function for the legal moves which gradually becomes more accurate throughout the training. Traditionally, state-transition functions are represented as $P(S'|S, A)$, where the function outputs the probability of transitioning into S' if action A is taken at state S. In a deterministic system, such as Othello, these transitions

are always certain. However, in a finite number of trials, the agent generally won't be 100% sure of anything, so probabilities are still used to represent how likely something is, in our case the probability that a position is a legal move. Since we are only interested in the legal moves, we can represent the state-transition function, P, in a much more compact way. Specifically, P now only takes one input, which is the state S, and maps S to a length 64 prediction vector K. Each index in K contains a_i, the probability of there being a legal move at location i. This is shown in (1).

$$P(S) = K = [a_0, a_1, \dots, a_{63}] \quad 1$$

To represent P, a Convolutional Neural Network (CNN) is used; its specific structure is described at the end of methods. This choice was made because CNNs have been shown to be suitable for RL problems (Sutton & Barto, 2018).

To interact with the environment in online training, the agent must also be able to decide what actions to make. To do this, the agent contains two policies, $\pi_d(S)$ and $\pi_s(S)$, which are the deterministic and stochastic policies, respectively. These policies take some state S as input and output action choice A. The policy, $\pi_d(S)$, is used in evaluation and the policy, $\pi_s(S)$, is used in training and evaluation. This choice is made to reduce overfitting, since $\pi_s(S)$ grants a bigger and more balanced distribution of action choices. We can formally define $\pi_d(S)$ and $\pi_s(S)$.

$$\pi_d(S) = A \ni P(S)[A] = max\big(P(S)\big) \quad 2$$

$$\pi_s(S) = A \ni p(A) = \frac{softmax\big(P(S)\big)[A]}{1} \quad 3$$

Formula (2) simply outputs the action with the most probability given by P(S). Formula (3) is a weighted choice, such that the probability $p(A)$ of choosing action A is $\mathrm{softmax}(P(S))[i]$. This makes it such that π_s allows a larger variety of moves for more exploration but still makes educated choices based on the agent's preferences.

Finally, the agent must be able to learn, done by improving P, based on the information it receives. This process is outlined in depth in the following sections.

Feedback and Training Return

A fundamental concept in RL is properly selecting the return scheme, which effectively dictates the feedback that the agent receives (Sutton & Barto, 2018). The simplest form of return presented in literature is discounted return, where the return is dictated by γ , $0 \leq \gamma \leq 1$, the discount rate. The higher γ is, the higher rewards in the future are valued. In this problem, it makes sense to choose $\gamma = 0$. This is because determining legal moves is greedy, having no consideration for the future.

Looking at it this way, the legal-move problem could also be labeled as having 1-step episodes, that being each state. For the purpose of this study, we consider episodes as entire games, rather than single states. This is because the agent ultimately explores entire episodes, and defining episodes this way makes it easier to discuss and frame the distribution problem. Regarding the learning process itself, $\gamma = 0$ entails that the return for a given position is always just the reward for that position; the reward is 1 if a position is a legal move or 0 if not.

Online Learning

During online training, the agent plays through many episodes. In each episode, the S, A, R, S', A' process described below repeats until the episode terminates.

1. The agent receives state S from the environment.
2. The agent chooses move A in an epsilon greedy manner with $\pi_s(S)$ and returns it back to the environment. Epsilon greedy is used to further ensure sufficient exploration of the environment.
3. The agent receives reward R based on the legality of A.
4. The agent receives the next state S', determined by playing move A on S. If A is illegal, the same state is returned back ($S' = S$). This extends the episode length.

Using the rewards gathered in the process above, we use back propagation as our update rule for the agent to learn. One could think of this update as leveraging temporal difference errors, however due to the one-step dynamics, it also makes sense to be seen as semi-supervised learning.

Back propagation is done through running training pairs through the CNN used to represent P. Each pair consists of state S and length 64 vector K. To make the training pair yield useful information, K starts as the current prediction $P(S)$ and then is modified with the rewards the agent receives when interacting with the environment. Specifically, each time the agent tries making move A, $K[A]$ is set to the reward returned. This process keeps occurring until S changes (when the agent plays a legal move). When this happens, the training pair is stored so it can be run as part of a batch later for faster runtime. Each batch consists of the training pairs from an entire episode.

Offline Learning

During offline training, the agent "experiences" sample games of Othello. Specifically, the agent receives a series of states and actions, S, A, S', A', which are the board states and actions that occurred in the sample game in order. From this information, the agent has to formulate an intelligent update rule to dictate its learning. We base this update rule off of state-action pairs, adjacent S and A received, as this tells the agent that A is a legal move at state S since A was used to successfully transition to S'.

The OPR problem emerges in offline learning. To reiterate, OPR comes from the fact that sample episodes only provide information on legal moves, that being A in each state-action pair. No information is given about other possible moves, including all illegal moves. It follows that it's trivial to give positive reinforcement, but difficult to give negative reinforcement to the agent. In this way, it does not suffice to update only A and leave the rest of the possible moves unchanged. We theorize that as there is no negative reinforcement, the agent will tend to think that all moves are legal with enough training. To verify this theory, the extent to which OPR is a problem will be tested experimentally using the following naïve update rule.

1. The prediction of the agent through P is saved; $P(S) = K$.
2. K' is set as a copy of K.
3. $K'[A]$ is set to 1, since A is guaranteed to be a legal move. All other positions are left unchanged.
4. An optimizer is run on P with training pair (S, K').

To overcome OPR, we propose using a method we call synthetic negative reinforcement. This method comes from the following two observations. First, there needs to be some sort of negative reinforcement to balance out the positive reinforcement such that the P does not converge to produce only 0s or 1s (Observation 1). Second, given optimal state-transition function P^*, there will be no effect of applying the update rule on P^*: $\text{update}(P^*) = P^*$ (Observation 2).

This leads to the following step being inserted between steps 3 and 4 of the naïve update rule:

$$d_i = \mathrm{K'[i]} * (1 - K'[i]) \; \forall \; \mathrm{i} \tag{4}$$

$$\mathrm{f_i} = |K[A] - K'[A]| \; * \; \frac{\mathrm{d_i}}{\sum_{j=0}^{63} d_j} \; \forall \; \mathrm{i} \; \ni \; \sum_{i=0}^{63} \mathrm{f_i} = |K[A] - K'[A]| \tag{5}$$

$$K'[i] = K[i] - \mathrm{f_i} \; \forall \; i \tag{6}$$

First, we want to consider how to distribute negative reinforcement. We do this through (4) which leverages the agent's previous experience to create d_i, the distributed negative reinforcement. This formula distributes more negative reinforcement to possible moves with predictions closer to 50%. This makes sense if we assume the agent is already somewhat educated in its predictions since we should apply more reinforcement to the uncertain predictions. In this way, Observation 2 is addressed; if the agent predicts exactly correctly, which happens in an optimal state-transition function, there will be no synthetic negative reinforcement produced. Observation 1 is addressed by (5), which creates scaled negative reinforcement $\mathrm{f_i}$ to ensure that the sum of all negative reinforcement balances out the positive reinforcement provided. Formula (6) applies the synthetic negative reinforcement.

Testing

Testing is done with agents trained on N offline episodes and M online episodes. Different pairs of (N, M) are tested such that $\mathrm{N} + \mathrm{M}$ is constant across all tests. This is so that using offline episodes doesn't give an intrinsic advantage of more episodes of total training. Later, these pairs are also written as $N + M$, where N and M are represented using "k" as an abbreviation for 1,000. The data for offline training is split into three categories: random, low-index skewed, and human played games. Only one type of data is used for offline training in a given trial.

Two benchmarks are used for comparison: random and supervised learning. Random is an agent with no training and supervised learning is an agent trained on fully labeled episodes for the same number of iterations. The supervised learning agent uses the same general process as the other offline agents. Note that the act of fully labeling episodes uses external information not available during offline and online training.

To gauge the accuracy of an agent, the agent plays through 100 unseen random offline episodes. Random episodes are used since we are interested in an accurate distribution for testing. Four metrics are recorded to gauge the accuracy of P, π_d, and π_s. These metrics are Mean Squared Error on P (MSE), Rounded Error on P (RE), π_d goofs (DG), and π_s goofs (SG). MSE is calculated through the standard process. Rounded error is calculated by rounding each index of the prediction to 1 or 0 and summing the number of positions the agent predicted incorrectly. Using rounded error, we can also calculate percentage accuracy by dividing the rounded error by the total number of positions. A "goof" is when a certain policy returns an illegal move at a state. The total goof count is tallied for each policy, yielding the average goofs per episode.

$$\text{combined} = \frac{MSE_{current}}{MSE_{supervised@end}} + \frac{RE_{current}}{RE_{supervised@end}} + \frac{DG_{current}}{DG_{supervised@end}} + \frac{SG_{current}}{SG_{supervised@end}} \tag{7}$$

To provide one metric to summarize all four for easier comparison, we create a 5th combined metric through (7), where "current" represents the metric at the current moment for some agent and "supervised@end" represents the metric of the supervised learning agent after all episodes. The supervised learning agent is used as a comparison because it best represents the most optimal result.

Specifications

The CNN used to represent P consists of two convolutional layers, one pooling layer, and 4 fully connected layers as illustrated in Figure 1 below. Note that the visual layer size is not exactly proportional to labeled layer size in figure. A sigmoid function is applied at the end to ensure all values are from 0 to 1 since they should be probabilities. The back propagation uses a learning rate of 10^{-3} with the Adam optimizer with Mean Squared Error (MSE) loss.

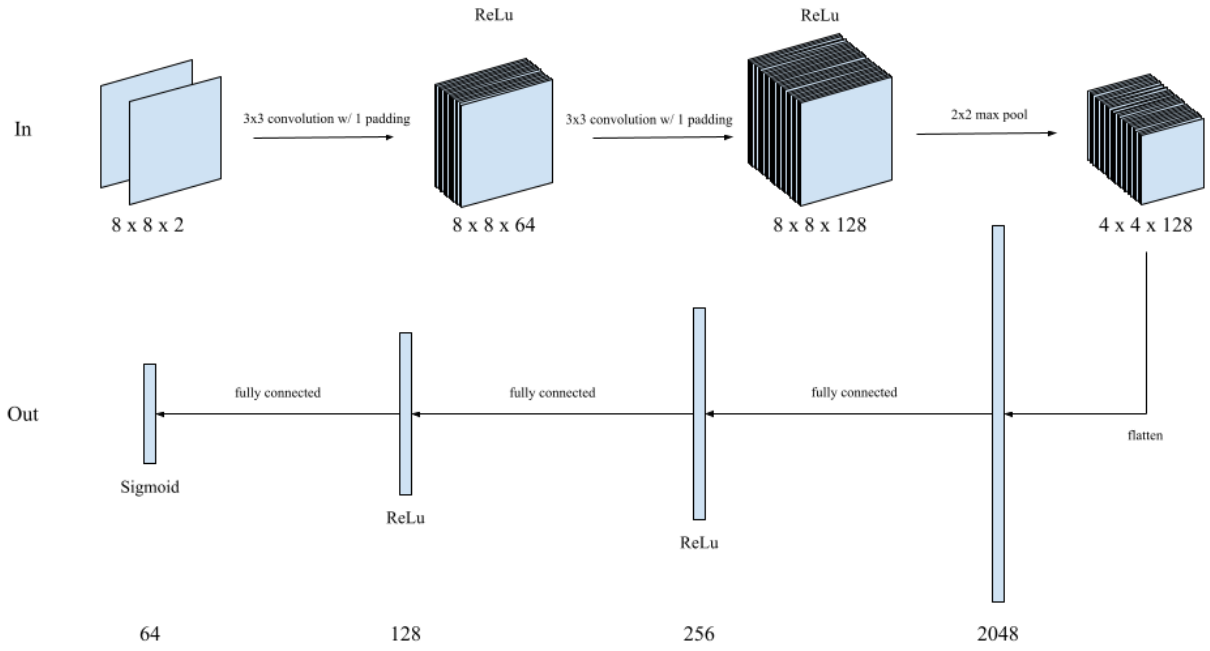

Figure 1. CNN Structure of $P(S)$

Python 3.10.12 and PyTorch 2.0.1+cu118 were used for the implementation of the presented methods. Training and testing were run on one GPU. All code to recreate the results presented in this paper can be found at [redacted].

Results

Table 1 below shows the metrics of the benchmark agents and agents with different combinations of offline and online training after 50,000 combined episodes ($N + M = 50{,}000$).

Table 1. Metrics of Different Agents. The best results (not including results from supervised learning) are shown in bold font.

Combination (offline*epochs + online)	MSE Error (Per state)	Rounded Error (Per state)	Rounded Accuracy (Per state)	Stochastic Goofs (Per game)	Deterministic Goofs (Per game)
No training	15.01	29.96	53.19%	50.83	49.81
No negative reinforcement	48.77	49.01	23.42%	50.85	46.94
0 + 50,000	1.54	2.04	96.81%	15.69	0.32

10,000*1 + 40,000	1.18	1.57	97.55%	14.61	**0.07**
25,000*1 + 25,000	1.24	1.65	97.42%	15.21	0.11
40,000*1 + 10,000	1.32	1.78	97.22%	**13.61**	0.10
50,000*1 + 0	2.59	3.32	94.81%	19.96	0.49
1000*10 + 40,000	**1.08**	**1.42**	**97.78%**	13.91	**0.07**
Supervised learning	0.59	0.76	98.81%	5.69	0.03

Agents using a combination of offline and online training are labeled in the scheme "offline*epochs + online" where the first two numbers are the offline episodes multiplied by the epochs, and the second number is the online episodes. Results for each metric is the average across 100 test episodes. Besides the benchmarks, the agents with only one type of training, 0 + 50,000 and 50,000 + 0, performed notably worse than the agents with hybrid training schemes (those using both online and offline learning). Within the hybrid agents, there is a general trend of the agents with less offline episodes having higher performance.

Table 1 also shows the performance of an agent without synthetic negative reinforcement following the naïve offline training method provided earlier. Only 10,000 episodes were run because no sign of improvement was shown in all first 10,000 episodes. After 10,000 episodes, the results of this agent were worse than the random agent with no training. On further inspection of the output matrices, this agent did in fact falsely predict that every square was a legal move.

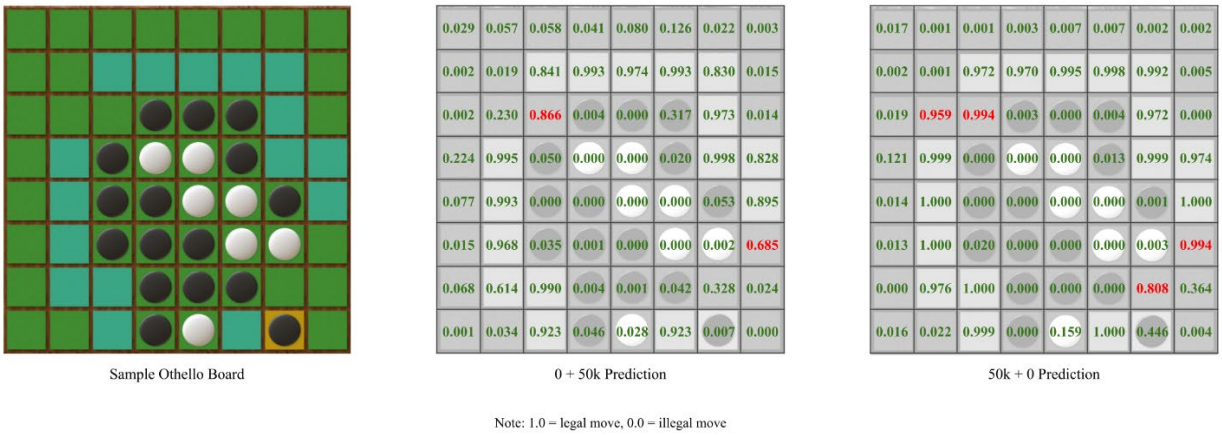

Figure 2. Example Prediction Visualization.

An arbitrary midgame Othello board with two agents' predictions is shown in Figure 2 above. On the sample Othello board (left), cyan squares represent legal moves while green squares and squares with tokens are illegal moves. In the prediction boards (middle and right), the numbers in the grids represent the probability that the agents think a certain square is a legal move. Red text shows a wrong prediction and green text shows a correct prediction. It can be noted that a distinct difference between the two agents' predictions is that the offline agent (50k + 0) is much more confident in all of its predictions, even when it is wrong. For example, this occurs in the third row from the top on the offline prediction board where the agent is around 96% and 99% confident that there is a legal move when in reality there is not. Meanwhile, the online agent (0 + 50k) is a lot more unsure with much more predictions close to 0.5.

The combined learning rates are plotted against each other as shown in Figure 3 below. The vertical axis measures the log_2 of the combined learning metric. This is done to scale the data for better visualization. Dots are marked where offline training stopped and online training began and thus are present only on the learning curves of hybrid agents. Datapoints are taken every 1,000 episodes. It can be seen that both the offline

and online training are effective, decreasing the combined learning metric. Before the online learning started, agents using offline learning performed significantly worse than agents already using online learning. At each dot, where the offline learning stops and the online learning kicks in, there is a drastic improvement. In all three hybrid examples, this drastic improvement is enough for the agent to surpass both the pure offline and pure online agents. All combination agents outperform the random no training agent and underperform the supervised learning agent.

The 10,000*1 + 40,000 and 1,000*10 + 40,000 agents have almost identical learning curves. This is interesting because, in theory, using 1,000 unique episodes over 10 epochs would be a much more constrained distribution than using 10,000 unique episodes over 1 epoch.

Only one training trial is used for each agent to generate the tables and due to the long training time of each agent. However, it is important to note that the other unrecorded and test trials, those not shown through figures or tables, conform to the trends we present here.

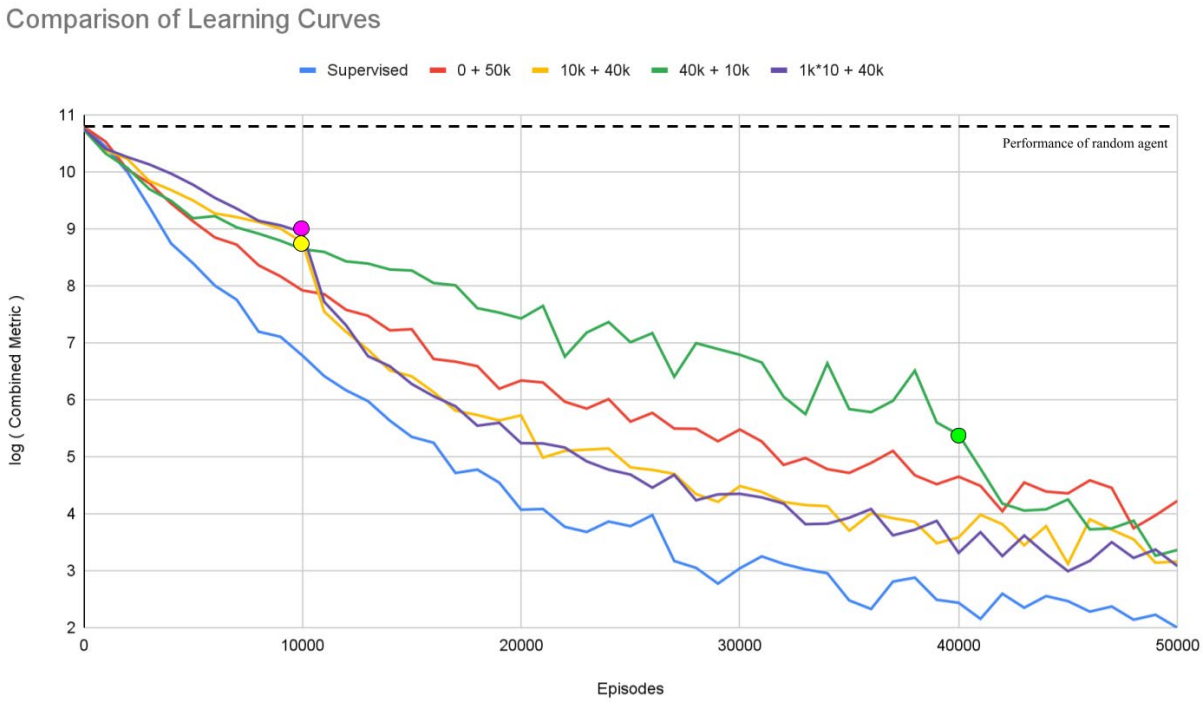

Figure 3. Comparison of Learning Curves.

Table 2 and Figure 4 highlight the results of conducting offline training using the skewed distribution of human played games. For the combination agent trained on the human distribution, the first 10,000 episodes are offline and from the human dataset. The remaining 40,000 are online using the standard process. In Table 2, all metrics are the average across 100 test episodes. In Figure 4, the vertical axis on both plots measures the log_2 of the combined learning metric. Dots are marked where offline training stopped and online training began.

Table 2. Metrics of Human Agent (10k + 40k) for Different Episode Counts.

Episodes	MSE Error (Per state)	Rounded Error (Per state)	Rounded Accuracy (Per state)	Stochastic Goofs (Per game)	Deterministic Goofs (Per game)
0	15.05	30.84	51.81%	51.78	49.4
10,000	7.97	7.97	87.55%	44.17	43.41
50,000	1.43	1.86	97.09%	15.45	0.18

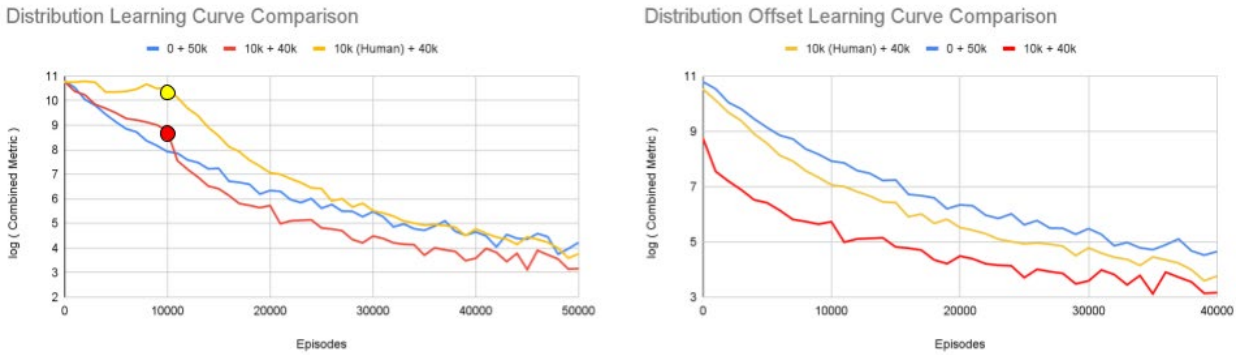

Figure 4. Comparison of Learning Curves for Different Distributions.

In Figure 4, the left plot shows a standard comparison, similar to Figure 3, while the right plot is offset such that each agent starts their online training at the 0-episode mark. It can be seen on the left plot that offline training on the skewed human distribution is generally not quite effective as the combined metric barely decreases. This same sentiment is echoed by Table 2 which shows that after the 10,000 episodes of offline training, the stochastic and deterministic goofs still remained very high. Although not displayed through any tables or figures, the same result occurred in agents using the self-generated distributions skewed by low-index.

However, these skewed distributions seem to not hinder the overall performance either. The left plot suggests that the human distribution model ended up performing around the same as the completely online training model, if not slightly better. The right plot, which is offset to compare the online sample efficiency, shows that the online sample efficiency after the human distribution training was not negatively affected. Perhaps, it was even slightly better since the difference between it and the 0 + 50k agent increased throughout the online training.

Discussion

As expected, the offline agent without synthetic negative reinforcement performed poorly. As shown by the results in Table 1, the metrics of this agent were worse than the random agent. This is due to the fact that the agent converged to produce all 1s due to the lack of negative reinforcement, which verifies our theory.

In the testing of the different combinations of offline and online training, we were surprised that the hybrid agents performed significantly better than the agents with only one training method. In this way, hybrid agents actually had higher sample efficiency. This is unexpected and counterintuitive as online learning provides more targeted feedback as well as just a higher quantity of information. The feedback is more targeted since whatever feedback received by the agent is directly correlated to P due to the setup of the π_s. In this way, the agent will receive more pointers on what specifically is wrong with its current “understanding." The higher

quantity of information is because choosing illegal moves extends the episode length which in turn generates more feedback. Since the number of episodes is held constant, the longer online episodes mean that online learning will always provide more information compared to offline learning.

Taking a look at the combined metric learning curves, we can see that indeed online learning is advantageous. In fact, the agents using online learning always significantly outperforms the agents that have been only using offline learning. The surprising results witnessed in the hybrid agents earlier only appeared after the agents switched to online learning. Immediately after the switch, a significant improvement is experienced and leads the hybrid agents to even surpass the online agent. This is interesting as the sample efficiency of the actual offline training was worse, but somehow it made the sample efficiency of the following online training more effective.

The most probable reason for this is the distribution problem. Our suspicion is that the offline training yielded the extra improvement on top of the online agent because the offline training provided more accurate information on the real distribution. Specifically, the hybrid agents that outperformed the online agent were trained on offline episodes from the random distribution, which best encompasses the environment in general and also is the testing distribution. This suspicion is supported by the results of the agents training on the skewed distributions, since those agents not only did not reap the same benefits as the random distribution agents, but also generally did not have successful offline training at all. This makes sense as the human distribution is very constrained on moves, because there are standard openings and obvious good moves. Many other moves are seldom played, and thus the agent while training on that dataset has no information on the majority of the state space. The same issue happens with the low-index distribution. Perhaps this issue could be overcome with a more optimal version of synthetic negative reinforcement.

On the bright side, these results show that with a robust distribution offline learning using our proposed synthetic negative reinforcement is certainly effective in terms of reducing the episodes needed of online learning. This is even true in the 1,000*10 epochs offline training, suggesting that there is much more importance in the distribution than the sheer quantity of available offline episodes. By robust, we mean distributions where games have reasonable chances of playing any legal move.

With this robustness constraint, one could perhaps completely replace the first chunk of needed episodes with offline episodes, effectively allowing the agent to leverage cheaper data while also having the chance to benefit from the distribution of that offline dataset.

As a final note, in comparison to the supervised learning agent, even the best hybrid agents performed around twice as worse according to the metrics in Table 1. While these results seem poor at first, we believe it's actually very impressive. The supervised learning agent is provided with feedback on all 64 positions, which is 64x more information than in offline learning and a significant portion larger than in online learning. In other words, the supervised learning agent is using a large amount of information inaccessible to all the other agents.

Future Work

Synthetic Negative Reinforcement: While synthetic negative reinforcement has produced reliable results with robust distributions, there is much room for improvement. As seen in the sample results from Figure 2, there are problems with the overconfidence of the offline agent due to the way that the formula for d_i is set up. Thus, testing should be done with different formulas for d_i to distribute the negative reinforcement. This could help with the distribution problem as well. Furthermore, the way we balance the positive and negative reinforcement is through a raw difference of the positive reinforcement. However, since the agent uses MSE loss, perhaps it would be wiser to use a different type of balancing.

Validation of Distribution Theory: In this study, we theorize that the reason why the hybrid agents outperformed the online agent was because of the robust offline distribution used, in this case random games.

This could be tested through having the online agent receive feedback on selected moves, but having the episode progress randomly (the next move is selected randomly from all legal moves).

Increasing Task Complexity: The task used in this study, finding legal moves, has one-step dynamics and is not too complex. To better gauge the efficacy of these methods, testing the model-learning aspect in environments where the results are not immediate would be wise. Also, different games could be tested besides Othello with different rulesets.

Post Model-Learning: We only looked at model learning. We hope to conduct performance training off of model-learning, effectively doing full MBRL. For example, a next step could be to develop a competent Othello playing program built off of a learned model of the environment. This will also help increase the task complexity and introduce longer environment dynamics.

Conclusion

To investigate offline model-based reinforcement learning, we tested agents trained on different combinations of offline and online learning through learning the legal moves of Othello. During this process, we encountered the problem where offline games provide Only Positive Reinforcement (OPR) and we propose a method using synthetic negative reinforcement to deal with this issue. This method leverages the agent's prediction to provide artificial negative reinforcement to prevent the state-transition function from converging to a state where all moves are thought to be legal.

Using a robust distribution of offline games, our hybrid agents (agents using first offline then online training) using synthetic negative reinforcement significantly outperform the agent using only online learning, with the best hybrid agent having 97.78% accuracy, which is almost 3% higher than the pure online agent. We suspect the additional benefit gained from using offline learning is due to the fact that the distribution of the initial offline dataset is an exact representation of the real environment. This fact is supported by the poor performance of offline training from human games and low-index skewed games, which are games from skewed distributions.

Moving forward, we intend to refine our method through optimizing details within the creation of synthetic negative reinforcement as well as expanding our scope to account for more challenging and robust tasks.

Acknowledgments

I would like to thank my advisor for providing insightful resources and feedback for my research. I would also like to thank my mentor, Ms. Joanna Gilberti, for helping me review my paper. Furthermore, I would like to thank my advisor again for introducing me to Ms. Shreya Sharma who provided valuable topic-specific feedback.

Finally, I'm immensely grateful for all of my teachers in the computer systems lab at school, especially my AI teacher, who got me interested in reinforcement learning in the first place.

References

Agarwal, R., Kumar, A., Zhou, W., Rajeswaran, A., Tucker, G., & Precup, D. (2022, December 2). *3rd offline RL workshop: Offline RL as a "Launchpad."* NeurIPS. Retrieved August 10, 2023, from https://offline-rl-neurips.github.io/2022/

Allis, V. L. (1994). *Searching for solutions in games and artificial intelligence* [Doctoral dissertation, University of Limburg]. François Grieu. http://fragrieu.free.fr/SearchingForSolutions.pdf

Buro, M. (2003). The evolution of strong Othello programs. *Entertainment Computing*. https://doi.org/10.1007/978-0-387-35660-0_65

Fédération Française d'Othello. (2023, April 26). *La base WTHOR*. Fédération Française d'Othello. Retrieved August 10, 2023, from https://www.ffothello.org/informatique/la-base-wthor/

He, Q., Liu, J. L., Eschapasse, L., Beveridge, E. H., & Brown, T. I. (2022). A comparison of reinforcement learning models of human spatial navigation. *Scientific Reports*, *12*(1), 1-11. https://doi.org/10.1038/s41598-022-18245-1

István, M. (2017). *How should I handle invalid actions (when using REINFORCE)?* [Online forum post]. StackExchange. https://ai.stackexchange.com/questions/2980/how-should-i-handle-invalid-actions-when-using-reinforce

Kidambi, R., Rajeswaran, A., Netrapalli, P., & Joachims, T. (2020). MOReL: Model-based offline reinforcement learning. *NeurIPS*. https://doi.org/10.48550/arXiv.2005.05951

Kim, Y., Lee, H., & Ryu, J. (2022). Learning an accurate state transition dynamics model by fitting both a function and its derivative. *IEEE Access*, *10*, 44248-44258. https://doi.org/10.1109/ACCESS.2022.3169798

LeCun, Y. (2022, June 27). *A path towards autonomous machine intelligence*. OpenReview. Retrieved August 8, 2023, from https://openreview.net/pdf?id=BZ5a1r-kVsf

Levine, S., Kumar, A., Tucker, G., & Fu, J. (2020, May 4). *Offline reinforcement learning: Tutorial, review, and perspectives on open problems*. arXiv. Retrieved August 1, 2023, from https://doi.org/10.48550/arXiv.2005.01643

Merkel, O. (2023). *UCThello*. GitHub. Retrieved August 8, 2023, from http://omerkel.github.io/UCThello/html5/src/

Quin, S., & Nicolet, S. (n.d.). *Format de la base Wthor* [Wthor database format]. Fédération Française d'Othello. Retrieved August 10, 2023, from https://www.ffothello.org/wthor/Format_WThor.pdf

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press. https://www.andrew.cmu.edu/course/10-703/textbook/BartoSutton.pdf

van der Ree, M., & Wiering, M. (2013). Reinforcement learning in the game of Othello: Learning against a fixed opponent and learning from self-play. *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*. https://doi.org/10.1109/ADPRL.2013.6614996

Wang, Z., Fu, Q., Chen, J., Wang, Y., & Lu, Y. (2023). Reinforcement learning in few-shot scenarios: A survey [Abstract from ProQuest Databases]. *Journal of Grid Computing*, *21*(2). https://doi.org/10.1007/s10723-023-09663-0

Xiang, S. (2021, December 19). *Reinforcement learning Q-learning with illegal actions from scratch.* Towards Data Science. Retrieved August 10, 2023, from https://towardsdatascience.com/reinforcement-learning-q-learning-with-illegal-actions-from-scratch-19759146c8bf