WHAT-IF: A Tool to Measure the Markov Property in Data Driven Reinforcement Learning

Jeremy Curuksu
People Experience and Technology
curukj@amazon.com

Charles Prosper
Amazon Web Services
rtpo@amazon.com

Abstract

Offline reinforcement learning can learn a policy from historical data collected from a Markov Decision Process. But when using historical data for offline learning, the reinforcement learning problem setup may not be Markov and the historical data may not have sufficient variability in states and actions to learn an optimal policy. In this paper, we propose a tool specifically designed to determine if some available historical data is created by a Markov Decision Process, and to measure the order of this Markov Decision Process. Our tool, which we call WHAT-IF, provides a first, purely data-driven method to measure the Markov Property in offline Reinforcement Learning. Combined with a GPT-based simulation framework to produce new agent-environment interactions based on historical data, we show that learning from data augmented by GPT improves the Markov property and overall quality of offline reinforcement learning, with up to 25% increase in accumulated rewards in typical optimal control problems.

1 Introduction

Reinforcement Learning (RL) methods can learn optimal decisions from limited data and/or feedback in order to reach a preset and potentially delayed goal, thanks to its ability to interact with an environment and learn by trial-and-error [1]. A fundamental problem that RL addresses, and which does not occur in supervised learning, is to balance exploration of new information with exploitation of the accumulated information to maximize reward. It is important to explore as well as to exploit, and many techniques exist to address this problem efficiently. Much less is known on assessing and addressing problems due to the underlying Markov assumption of the RL explore-exploit loop in sequential decision-making processes [2]. Most RL methods assume the environment is a fully- or partially-observable Markov Decision Process (MDP), meaning the information available in any state, regardless of the past trajectory leading up to this state, is sufficient to make decisions about future states and future reward [1]. While this assumption is often at least partially violated and leads to deterioration of RL performance, there is no available tool nor consensus on how to assess the extent to which a RL problem is Markov. Despite these limitations, advances have been made in RL applications such as video games [3], board games [4], robotics, autonomous driving, and many others, by leveraging offline (simulation) environments and combining RL with deep learning function approximation [3, 4]. In offline RL, a policy is learned from a static dataset to take advantage of large, previously-collected datasets. Recent effort producing the first benchmarks designed for the offline setting and relevant to real-world applications [2] have further emphasized the need to measure the Markov property, but provided no solution yet to meet this need. Function approximation with deep neural network, on the other hand, is able to generalize through very large state-action spaces and thus reduce the reliance of RL on the Markov property. Still, there is no tool nor consensus to measure the Markov property and thus to understand and address this problem, specifically.

In this paper, we propose a tool specifically designed to measure the Markov property in offline and sequential RL setting, which uses a modified version of the concept of information gain [5] as a

function of lags (e.g., time step in temporal RL settings) to test for multiple MDP properties. Our tool, which we call WHAT-IF, provides a first, purely data-driven method to measure how much actions affect future states, and how much rewards depend on actions taken in a given state. We show that WHAT-IF can determine the order of a MDP directly from an available dataset. Combined with a GPT-based simulation framework to generate new sequences based on historical data [6], we show that learning from data augmented by GPT improves the Markov property and overall quality of offline RL. This is measured on a typical optimal control problem (indoor temperature control environment) where the GPT-based offline data lead to 25% increase in learning performance.

2 Methodology

A key requirement for RL data defined as a sequence of tuples (*state, action, reward, next state*) to be collected according to a Markov Decision Process (MDP), is that actions affect future states and future rewards [1]. In particular, rewards must depend on actions taken in specific states for the agent to be able to learn. The MDP can be explored explicitly in an online setting where the agent directly interacts with the environment, but when working with offline data, the data is often biased by the original data generation process which the agent did not control [2]. There is no guarantee the data was generated by a MDP. For example, data recorded on a trash-collector robot which was systematically bumping into a wall and received reward only when manually taken out of this kinetic trap may not cover the action space of interest. When trained on this data, the agent could learn that some actions away from the wall lead to a reward, not because these are optimal actions for collecting trash but because pulling away from the wall is the only event ever sampled in this dataset.

To measure the relationship between the actions taken and the rewards received in future states, and the overall order of the MDP, we use a modified version of the concept of information gain [5] as a function of lags and derive lag-dependent measures directly from the data. This contrasts with the only other method published, to our knowledge, which tests the Markov assumption in reinforcement learning [7]. The method in [7] uses an iterative approach based on double Machine Learning to simultaneously predict forward and backward correlations among observed states, actions, and rewards, until convergence to the covariance matrix that best fits the dataset. Due to its iterative procedure, this method can converge on small data sets [7] but not on large datasets, as confirmed by experiments we made ourselves on large datasets.

The method we propose, which we call WHAT-IF, uses a process of quantization of all observed states, actions and rewards available in a starting RL dataset, organized within groups defined by similar regions of the state-action space. Groups are defined by a future state and/or future reward. This is followed by randomization of each quantized state and action within each group, so as to measure the loss in entropy (information gain, [5]) induced by taking specific actions in specific states for each group i.e., for a given outcome defined by a future state or reward.

2.1 Problem Statement

For an agent to learn by reinforcement offline, there should be a significant benefit for the agent to know how actions influence future states and rewards in the past. In other words, the temporal covariance between any single past state and action with *some* future state(s) and reward(s) must be significantly different from a random process. This is known as the Markov Property, which implies that once the current state is known, the past trajectory that led up to it may be thrown away because the current state includes sufficient information to make decisions about the future. The degree to which this property is valid depends on the aforementioned temporal covariance as well as on how much is observed about the environment in cases of *partially observable* MDP:

$$p(s, r \mid s^t, a^t) = p(s, r \mid h^t, a^t)$$

where h^t is the entire history up to time *t*. The method in [7] for testing this property fits a model to predict future states from past states and actions. This supervised learning approach predicts the

likelihood or expected values of future states and rewards based on a quadratic loss function assuming the existence of a stationary MDP. It doesn't directly address the Markov property i.e., it doesn't directly measure the relationship between past actions and future states and rewards, which leads to bias or variance issues (i.e., convergence issues) whenever the Markov assumption is only partially valid. The method we propose does not use supervised learning and instead directly measures the information gain/entropy loss resulting from taking a specific action in a specific state.

Step 1: Quantization of RL tuples (state, action, reward, next state)

We use a non-linear quantile transformation to estimate and discretize a complete cumulative distribution function for each action taken in each state mapped to a given outcome, either a next state or reward. This can also be mapped to any number of steps in the future, which we will use to go beyond nearest-neighbor effects and measure the *order* of the MDP. Depending on the size of the dataset, this transformation of observed states and actions toward uniform distributions mapped to specific future states or reward can be fine-tuned to be more or less precise by using a different number of quantiles. More details on the quantile transformation method can be found in [8].

One drawback of this quantization is it reduces the impact of outliers and the Markov assumption itself, but this drawback is by far outweighed by the benefits of its model-free estimation approach. It defines a purely data driven, non-parametric function leveraging all information available to define tokens of states and actions, grouped as per their influence on future states and reward.

Step 2: Measure Entropy in RL tuples (state, action, reward, next state)

The Shannon entropy [5] is measured separately in each quantized state and action as defined in the previous section, and also in the groups to which these state-action tokens belong defined by either a future state or reward n steps in the future, after randomly shuffling the corresponding state or action within each group. We tested the order of the MDP up to 9 i.e., $n = 1$ to 9.

This method enables us to measure the *loss* in entropy (information *gain*) for each group i.e., for a given outcome defined by a future state or reward, induced by taking specific actions in specific states. If some information is gained (i.e., significant loss of entropy) when comparing the entropy conditioned on a particular action taken in a particular state vs. a randomly shuffled set of actions or states, then we assume a relationship exists between that particular action or state and the observed future outcome. Again, the conditioning variable can be lagged to test for the effect of actions multiple steps in the future i.e., to assess the order of the MDP.

In practice, groups defined by future states/rewards contain several tokens and thus comparing the entropy of any given token with the entropy of the group always leads to a reduction in entropy, in particular after marginalization within group by shuffling tokens randomly, as proposed here. The relative reduction between groups informs us on the significance of the Markov assumption between actions and outcomes in the available dataset

Step 2.1: Measure Controllability and Influenceability

We used the approach described above to measure the information gain separately for (1) the future states as a function of the action taken in the current state (current section), (2) the future rewards as a function of the action taken in the current state, and (3) the q-value of the action taken in the current state. To measure how much the action taken in the current state influences future states, we measure the following information gain [5]:

$$I(s^{t+1} \mid a^t) = H\left(s^{t+1} \,\middle|\, s^t, \text{rand}(a^t)\right) - H(r^{t+1} \mid s^t, a^t)$$

where for any variable x, $H(x)$ is the entropy of x as defined above [5]. This measure informs us on whether future states are controllable and influenceable, and not purely exogenous factors out of the agent's control.

Step 2.2: Measure Latent Reward Function

To measure how much the action taken in the current state influences future rewards, we measure the following information gain:

$$I(r^{t+1} \mid s^t,\, a^t) = H\left(r^{t+1} \,\middle|\, s^t, \mathrm{rand}(a^t)\right) - H(r^{t+1} \mid s^t, a^t)$$

This measure informs us on whether to assign credit to actions taken in current state given the reward received. We tested delayed reward with n = 1 to 9 where n is the number of steps in the future.

Step 2.3: Measure Q-value estimates

Finally, to measure how much the action taken in a state influence its Q-value estimate, we estimated the n-step TD action values using the on-policy SARSA algorithm, resulting in a Q-table discretized as per the quantile token defined above for the entire states-action space.

Step 3: Consolidate Results in a Markov Matrix

We consolidate all results above in a single 9x4 matrix containing nine columns corresponding to testing for n steps in the future, and four rows containing respectively the overall Markov order defined as the combined information gain from a given state and action on the next state (n steps in the future), the overall reward function defined as the combined information gain from a given state and action on the next reward, the action contribution defined as the information gain from a given action on the next reward, and the action effectiveness defined as the information gain from a given action on the next state. Example of Markov matrices are shown in the Results section.

GPT Simulator and Planner

To showcase the WHAT-IF method in practice, we also implemented a GPT-based simulation framework that leverages the Transformer technology [9] to produce new agent-environment interactions based on historical data. We then use WHAT-IF to show that learning from data augmented by GPT improves the Markov property and in turn, the overall quality of offline reinforcement learning. The GPT-based simulation framework was derived from the recently published sequence modeling method called Trajectory-Transformer [10], which we trained on the quantile tokens defined above which estimate a complete cumulative distribution function for each action taken in each state mapped to a given outcome (an "outcome" is defined as either a future state or a future reward). Our quantile tokenization helps extend and split continuous state-action spaces into uniform and discrete distributions as described above, and in turn ensures the synthetic data generated by GPT can sample as widely and uniformly as possible the entire state-action space extrapolated from the available dataset.

In [10], the original implementation of the Trajectory-Transformer method was solely focused on offline datasets generated by Gym environments, so by developing our quantile tokenizer we are extending the Trajectory-Transformer method to any custom, real-world RL data using a standardized and streamlined process. In particular, our custom GPT-based simulation framework based on Trajectory-Transformer directly applies to partially-observable MDPs.

3 Results

We evaluated WHAT-IF on two case studies, one using the Gym environment TAXI derived from [11] and one using a completely custom environment on indoor temperature control.

3.1 Gym Taxi-v3 Environment [11]

The Taxi-v3 environment provided by OpenAI Gym is a classic reinforcement learning problem in which an agent controls a taxi to pick up and drop off passengers at specified locations. The environment is represented as a grid-world with a 5x5 grid, where the taxi and passengers are

represented as different colored squares. The taxi can move in four directions (north, south, east, west) and can pick up or drop off passengers at four designated locations. The goal of the agent is to navigate the taxi to pick up and drop off passengers in the shortest amount of time possible while minimizing the number of moves. A Deep Deterministic Policy Gradient algorithm was used to train an RL agent for 10,000 steps of the Taxi-v3 Gym environment, setting an ε-greedy exploration schedule to $\varepsilon = 0.1$. Figure 1 shows the resulting Markov Matrix (detailed results for rows 1 and 2 are shown in Table 1). It indicates the relationship between actions taken vs. future states and reward is significant, and thus the Markov property is valid in this environment, with a strong Markov order 1 (green cell in Figure 1). This means the action taken in the current state has a strong influence on the nearest-next state, on average. The information gain for the next state is still significant up to 3 steps in the future (> 0.6), and even up to 9 steps in the future (> 0.6).

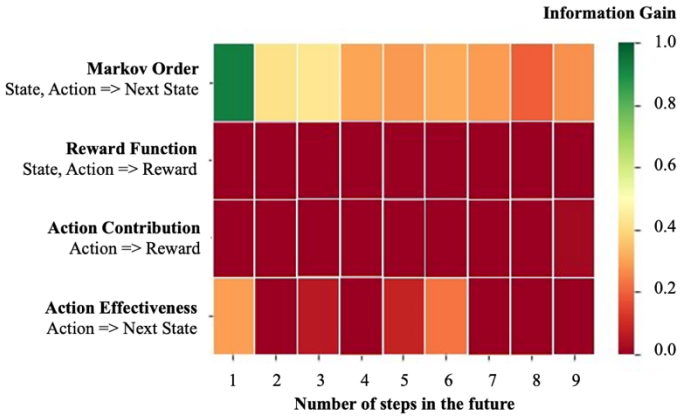

Figure 1: Markov Matrix showing the impact of current state and action on future state and reward up to 9 steps in the future for the Gym Taxi-v3 RL environment

Table 1: Subsection of the Markov Matrix shown in Fig. 1 (Taxi-v3)

	Number of steps in the future								
Information Gain (in Bits)	1	2	3	4	5	6	7	8	9
Markov Order	1.28	0.57	0.42	0.3	0.36	0.25	0.26	0.37	0.34
Reward Function	0.05	-0.08	0.02	0	0	0.14	-0.08	0	0.03

In contrast, the second and third rows of the Markov Matrix in Figure 1 indicate that the influence of actions taken on future reward is not significant in any of the 9 nearest-next steps. This can be explained by the very sparse (delayed) reward in the Gym Taxi-v3 environment, where reward is sent to the agent only upon termination of an episode, that is only when the taxi ultimately drops its passengers off at one of the four locations. Reaching the agent's goal most often takes more than 9 steps in our simulations. This explanation is further supported by the fourth row in Figure 1 which indicates a significant information gain from the action taken onto the Q-values, that is the influence of actions taken onto the expected accumulated reward predicted by the SARSA algorithm for the current and some of the nearest-neighbor states.

3.2 Custom Temperature Control Environment

We applied WHAT-IF on a custom optimal control problem, which here we refer to as the indoor *temperature control environment* and which was a partially observable MDP in the sense that external non-observable factors could influence the observed states and rewards. This custom temperature control environment was derived from an AWS customer problem and involves heating

several buildings in an urban district through a water-based, central heating system located under the floor and regulated by a district-wide central energy plant.

A dataset was built by collecting hourly measurements, over an entire season (6 months) on indoor and outdoor temperature and humidity, supply water temperature, and several additional factors that define the RL agent's state. The dataset also tracked values of the controllable parameter defining the RL agent's action, which was the setpoint for the underfloor heating system powered by water reserves. These water reserves transfer thermal energy by conduction to each room of a building, and are regulated by a central energy plant. The goal of the agent was to adjust the setpoint for the underfloor water returning to the central energy plant after it has been used for heating. This was a key customer problem because the efficiency of the heating system for entire districts depends on the temperature difference between the supply and the return water: if the return water is too hot, the heating demand of the building being served is not being met and more heat should have been transferred to this building; if the return water is too cold, too much heat has been transferred and the central energy plant may not be able to meet the demand of subsequent buildings. We used a custom reward function provided by the customer to calculate an empirical optimal setpoint value based on the agent state for sparse and predefined time points, and rewarded the agent accordingly based on the deviation of its actions from this sparse feedback, so the agent could learn to behave optimally, using sequential decision making from states with no reward feedback to states with available feedback to the extent that the environment was Markov. In the rest of this section, we use WHAT-IF to measure the extent to which this environment was Markov.

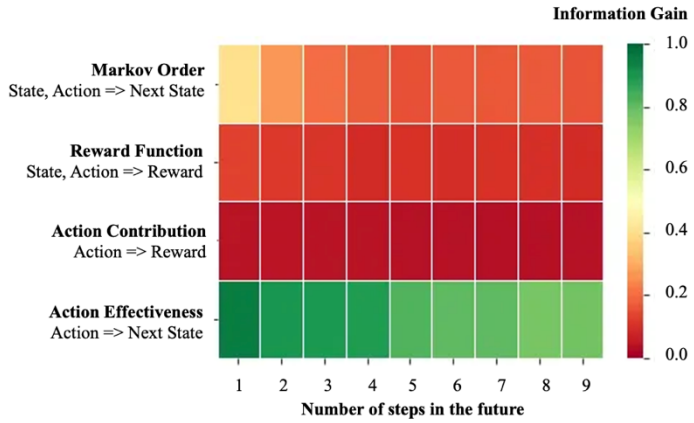

Figure 2: Markov Matrix showing the impact of current state and action on future state and reward up to 9 steps in the future for the custom temperature control RL environment

Table 2: Subsection of the Markov Matrix shown in Fig. 2 (temperature control)

	Number of steps in the future								
Information Gain (in Bits)	1	2	3	4	5	6	7	8	9
Markov Order	0.42	0.28	0.21	0.18	0.16	0.18	0.17	0.18	0.17
Reward Function	0.12	0.11	0.12	0.11	0.11	0.1	0.1	0.11	0.11

As can be seen on Figure 2 showing the resulting Markov Matrix for the temperature control environment (detailed results for rows 1 and 2 are shown in Table 2), the relationships between current state and action vs. future state and reward identified by WHAT-IF indicates that the Markov property is valid in this environment, with a significant Markov order 2 (> 0.3). This means the action taken in the current state has a strong influence on at least the two nearest-next states, on average. The information gain for other states is non-zero (> 0.1), thus potentially also significant.

Unlike in the above Gym Taxi-v3 dataset, a significant information gain is observed for actions taken in specific states on future reward (> 0.2), which confirms the validity of a Markov assumption in this temperature control problem. Interestingly, the information gain for actions irrespective of the state in which these actions are taken is relatively low on the future reward (< 0.2), and very high (> 0.8) on the nearest-next states. This illustrates a subtle but often overlooked impact of reward function design in RL: decisions made by the agent have a strong impact on all future states (up to order 9), but the information used by the agent to make these decisions (its "state") is insufficient for the agent to significantly contribute and control its environment. This is expected in partially-observed MDPs, and WHAT-IF provides a clear measurement of *how much* actions influence future states vs. contribute to solve the actual control problem i.e., contribute to reach the actual goal of the agent. Figure 3 summarizes some of the above results comparing the Markov Order and influence on the reward function for the Taxi-v3 vs. custom temperature control environments.

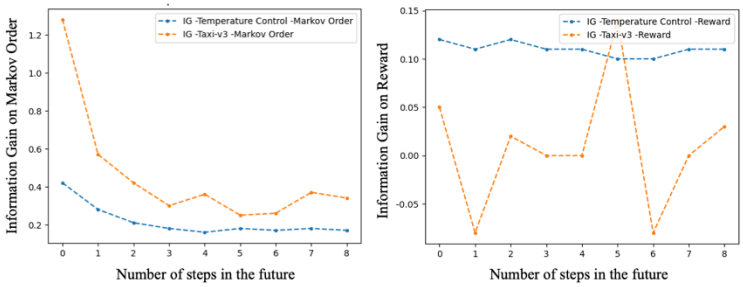

Figure 3: Comparison of Information Gain on the Markov Order and reward function between the Taxi-v3 and temperature control environments

We applied the GPT simulation framework based on Trajectory-Transformer described in the methodology section to this custom, partially observable temperature control MDP environment. The synthetic data generated by GPT can sample more widely and uniformly the entire state-action space extrapolated from the original dataset. We assess the results by measuring the average across all GPT simulations of the accumulated reward per time step in the future, and by comparing this distribution with the Markov Order. We also compare the results of this GPT simulation with a more standard offline RL method called Conservative Q-Learning (CQL, [12]) which directly learns from the original dataset (without the GPT synthetic data), to assess the efficiency of the GPT simulations. CQL addresses the off-policy problem by allowing for exploration of new states under some regularization and has shown some promises for offline RL using pre-existing datasets.

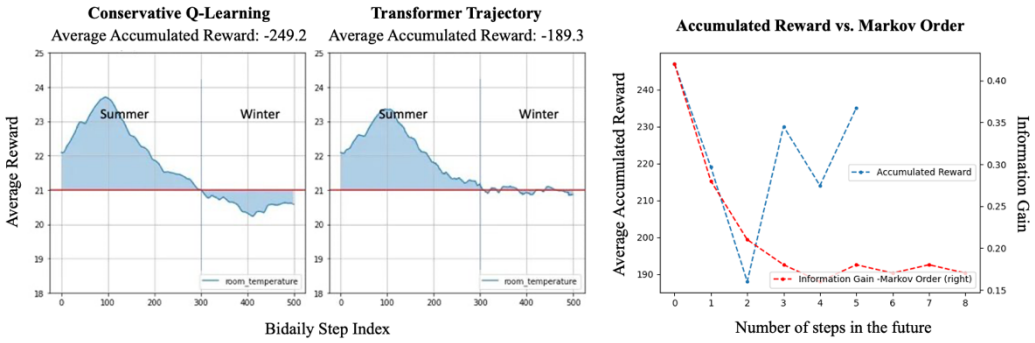

Figure 4: Accumulated reward per time step (twice per day) in the average trajectory simulated by CQL and GPT (left) and average accumulated reward/Markov Order (right)

As can be seen in Figure 4, we observed up to 25% increase (-189.3 vs. -249.2) in accumulated rewards when using additional data simulated by GPT vs. when using CQL on the original dataset. Most of these rewards tend to be received over the next two nearest-neighbor states on average, which coincides with the Markov Order 2 hypothesized above based on the original dataset (Figure

2, Table 2). Interestingly, the additional data simulated by GPT with Trajectory-Transformer led to a much clearer pattern of order 2 influence on the reward function compared to the original dataset: we now observe a peak in the average accumulated reward received two steps in the future, which is very apparent in Figure 4 (right figure).

4 Conclusion

We proposed WHAT-IF, a tool specifically designed to measure the Markov property in offline and sequential RL settings, which uses a modified version of the concept of *information gain* [5] as a function of lags (i.e., time step in temporal RL settings) to test for multiple MDP properties. WHAT-IF provides a first, purely data-driven method to measure how much actions affect future states, and how much rewards depend on actions taken in a given state.

In the Taxi-v3 Gym environment, the resulting Markov Matrix indicated that the Markov property is valid in this environment, with a strong Markov order 1. It also provided detailed insights into how much states and actions influence the future expected reward. For example, the information gain for the next state was still very significant up to 3 steps in the future (> 0.6).

In the custom temperature control environment, the resulting Markov Matrix indicated that the Markov property is valid in this environment, with a strong Markov order 2. It also showcased subtleties such as a very high dependence of future states on actions taken irrespective of the state in which these actions are taken (> 0.8), but comparatively very low dependence on the future reward (< 0.2). This illustrates typical challenges in partially-observable MDPs where information available is often too sparse for the agent to effectively control its environment. But WHAT-IF can provide, for the first time, a clear measurement of how much these actions influence or not the future states vs. contribute or not to solve the actual control problem that the agent is trying to solve.

Combined with a GPT-based simulation framework, we could show that planning from data augmented by GPT improves the Markov property and overall quality of offline RL, with up to 25% increase in accumulated rewards in the custom temperature control problem.

Future research will include improving the bound between the original and randomized set used in the quantile transformation required to compute the Markov Matrix, and increasing the scope of the overall entropy comparative analysis with new metrics such as perplexity. Overall, WHATIF offers a simple, effective, and purely data-driven method to measure the Markov Property in offline RL, including in dataset with sparse reward, and with GPT-based simulation frameworks it may offer a systematic method to improve RL planning and sample efficiency.

Customer Problem Statement

Reinforcement Learning has been very successful over the past few years in applications such as video games and board games where it reached definitive superhuman abilities and thus, has *come of age.* RL is now the object of active experimentation by Amazon and AWS customers in nearly all industries, most importantly supply chain optimization where RL has been successfully deployed online to optimize product selection at Amazon, robotics, autonomous driving, finance, energy, and many others. Offline RL promises to learn optimal RL policies from "historical" data collected by customers. But RL is often only as good as the data is "Markov", and there is no clear way to date to measure this Markov property and help customers build dataset from which an RL agent can learn efficiently offline. In this paper, we propose WHAT-IF, a tool specifically designed to measure this property and to make concrete RL design assessments on whether the states can be defined from an existing dataset pre-built by a customer, and whether the action the customer is seeking to optimize does influence the future states and future reward in this dataset. Also, WHAT-IF measures "how much" these actions influence future states and future rewards. In turn, WHAT-IF measures how valid and effective a given RL offline setup is for an agent to learn from a given, pre-existing dataset.

References

[1] Sutton RS, Barto AG (2018) Reinforcement learning: An introduction. MIT press: 47-141

[2] Fu J, Kumar A, Nachum O, Tucker G, Levine S (2020) D4rl: Datasets for deep data-driven reinforcement learning. arXiv preprint arXiv: 2004.07219

[3] Mnih V et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540): 529-533

[4] Silver D, Hubert T, Schrittwieser J et al. (2018) A general reinforcement learning algorithm that masters Chess, Shogi and Go through self-play. Science 362: 1140–1144

[5] Gorban A, Gorban P, Judge G (2010) Entropy: The Markov Ordering Approach. Entropy. 12(5):1145-1193

[6] https://github.com/awslabs/amazon-accessible-rl-sdk

[7] Shi C, Wan R, Song R, Lu W, Leng L (2020) Does the Markov Decision Process fit the data: testing for the Markov property in sequential decision making. In International Conference on Machine Learning, PMLR: 8807-8817

[8] Pedregosa et al. (2011) Scikit-learn: Machine Learning in Python, JMLR (12): 2825-2830

[9] Vaswani A, Shazeer N et al. (2017) Attention is all you need. Advances in neural information processing systems, 30: 5998-6008

[10] Janner M, Li Q, Levine S (2021) Offline reinforcement learning as one big sequence modeling problem. Advances in neural information processing systems, 34: 1273-1286

[11] Gym Taxi environment: Dietterich TG (2000) Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition, Journal of Artificial Intelligence Research, 13: 227–303

[12] Kumar A, Zhou A, Tucker G, Levine S (2020) Conservative Q-Learning for Offline Reinforcement Learning. arXiv preprint arXiv: 2006.04779