Performance of Decision Transformer in POMDP's

Nathan de Lara
McGill University
COMP 597

Abstract

This report shows the results of applying Decision Transformers to a Partially Observable Markov Decision Process. We convert the CartPole problem into a POMDP problem by obscuring the velocities in the state, we then evaluate different algorithms on the POMDP CartPole. We train one DDQN agent on the CartPole problem with full state information and another with the partially observed states and then evaluate the Decision Transformer when trained on a dataset of the fully observing DDQN, partially observing DDQN, and an equal mix of both agents. The performance of the Decision Transformer is then compared to Conservative Q-Learning (CQL), Offline DDQN, and Behaviour Cloning (BC) when trained on the same dataset. We find that Decision Transformer outperforms all the other algorithms on every dataset but is still unable to reach perfect results.

1 Introduction

Partially Observable Markov Decision Processes (POMDP), are processes in which the markov assumption is satisfied but the full state information is never observed by the agent. The canonical CartPole problem becomes POMDP when we remove the derivatives of the x-position and angle from the observation space. POMDPs are an important research area as in many applications of RL it is not possible to get full knowledge of a system. For example, when deciding dosage amounts when using chemotherapy to treat cancer, using too low a dosage will be unsuccessful in removing cancer cells, whereas, using too high a dosage will likely kill the cancer cells but will also greatly weaken the patients immune system and lead them to being susceptible to life-threatening infections. In this task, physicians never have complete knowledge as this would require knowledge about every cell in a patients body.

Reinforcement Learning is best done in an online setting where a policy π, parameterized by θ, performs actions in an environment and then updates θ by performing more actions in a constant loop. However, online RL is not always feasible due to the lack of a good simulator or ethical concerns with deploying an agent to explore and make mistakes, as is the case in medical applications. Hence, there is a need for offline RL algorithms that can learn good policies from data without an environment. If offline RL algorithms are not carefully defined they can suffer from the deadly triad, i.e., offline learning; bootstrapping; and value function approximation which can cause instabilities and divergences in the learning process. Current Actor-critic and Q-learning methods for online RL rely on bootstrapping and value approximation for their learning and so the common online RL algorithms are not well suited for offline RL and can have divergences when learning from a fixed dataset.

Transformers have outperformed RNNs and LSTMs in sequence-to-sequence problems, such as language modelling. Decision Transformers are another variant of Transformer based RL algorithms but for the offline setting. Decision Transformers have been shown to improve on or reach state of the art offline RL benchmarks in MuJoCo and Atari environments. Most importantly, Decision Transformers do not use bootstrapping in their learning process nor value approximation so they do not fall victim to the deadly triad. The success from applying Transformers to these sequential problems points to their ability to learn complex functions over spaces of sequences and non-markovian dependencies between elements in sequences.

This project aims to help fill in the gap in understanding how Decision Transformers behave in Partially Observing Markov Decision Processes be evaluating it in such a setting and comparing it to existing offline RL algorithms. By obscuring the velocities the CartPole problem becomes a POMDP, we train a partially observing DDQN agent and fully observing DDQN agent and then evaluate the Decision Transformer on a dataset with actions from the partially observing DDQN, actions from the fully observing DDQN, an equal distribution of actions from the two expert agents. We find that Decision Transformer outperforms Conservative Q-Learning (CQL), Offline DDQN and Behaviour Cloning (BC) on all three datasets but fails to achieve perfect performance. Our results point to the strength of Decision Transformers as an offline RL method but also suggest the need for improving the architecture or increasing the scale of the models if perfect performance is to be attained.

2 Background

We discussed the POMDP setting, offline RL setting, CartPole environment, and DDQN in class so the only parts of this project which were not covered are the Decision Transformer and Conservative Q-Learning algorithms. Here I will provide the necessary background on both.

2.1 Decision Transformer

The Decision Transformer is trained in a supervised fashion where it is given an incomplete trajectory $\tau = (R_1, s_1, a_1, \ldots R_T, s_T)$ missing only the last action a_T and the task is to predict a_T [Che+21]. In the trajectory the s_t are the states at their respective time t, the a_t are the actions chosen at the respective time t, and the R_t are the sum of future rewards of this trajectory, i.e. if r_t is the reward observed at time t, $R_t = \sum_{i=t+1}^{T} r_i$ [Che+21]. In discrete action spaces, cross-entropy loss is used to train the Decision Transformer, and in continuous action spaces mean squared error is applied. The architecture for the Decision Transformer is a GPT-decoder where the number of layers and number of heads in the multi-head attention block are task-specific, a more thorough explanation of Transformers can be found here [Vas+17]. For evaluating the Decision Transformer and using it to act in an environment R_1 is pre-specified and the following R_t are defined as $R_t = R_1 - \sum i = 1^t r_t$, the rest of the trajectory are the real state and action histories in that episode [Che+21]. the suggested practice is to set R_T to be above the maximum observed rewards-to-go in the dataset [Che+21].

2.2 Conservative Q-Learning

Conservative Q-Learning (CQL) is a state of the art algorithm for offline RL originally proposed to address overestimation [Kum+20]. Seno and Imai , provide an implementation for a variant of CQL which uses one additional regularizer as opposed to the original algorithm which used three [SI22]. CQL learns a conservative estimate of the Q-function by minimizing the DDQN error, and minimizing over all the state-action values while maximizing the state-action values for actions common in the dataset. This results in the following loss function:

$$\mathcal{L}(\theta) = \alpha(\mathbb{E}_{s_t \sim D}[log \sum_a expQ_\theta(s_t, a)] - \mathbb{E}_{s_t, a \sim D}[Q_\theta(s, a)]) + \mathcal{L}_{DDQN}(\theta)$$

By minimizing this loss, the Q-function minimizes state-action values for rare state-action pairs while maximizing the state-action values in the dataset constrained by the DDQN loss leading to conservative estimates of state-action values which are not common.

3 Methodology

An RL agent in the CartPole problem must make the decision between moving the cart left or right given the current x-axis location of the cart, horizontal velocity of the cart, angle of the pole with respect to the cart, and the angular velocity of the pole. By withholding the horizontal and angular velocities from the state space, the task becomes a POMDP. So, to evaluate the Decision Transformer on an offline POMDP task I evaluate the performance of the Decision Transformer on the CartPole task with the velocities removed from states. Since the Decision Transformer is an offline approach to RL, it's performance is ultimately dependent on the quality of the data and the policy used to generate that dataset. I evaluate the Decision Transformer across 3 data sets: (1) a dataset generated by a DDQN algorithm trained on the POMDP CartPole environment, (2) a dataset generated a DDQN algorithm trained on the CartPole environment with full observations available, (3) a dataset which is the mix of (1) and (2). I compare the performance of the Decision Transformer to Conservative Q-Learing (CQL), offline DDQN, and Behaviour Cloning (BC).

Generating the three datasets requires training two policies online. The first being a DDQN trained online with the POMDP states, and the second being a DDQN trained online with the fully observed states. For training the fully observing DDQN, adding a reward of -100 at terminal states led to faster convergence towards the perfect policy but the same impact was not observed when the change was made for the POMDP DDQN. The POMDP DDQN was unable to attain perfect performance while the DDQN with fully observed states was. Because the POMDP DDQN never achieved perfect performance its episodes are significantly shorter than episodes under the fully observing DDQN, so to have the number of transitions in both datasets be equal, 5000 testing episodes were used to collect data from the POMDP DDQN and 1000 episodes were used for the fully observing DDQN.

To train and test the Decision Transformers, I used the model architecture described in Appendix A combined with a single linear layer at the end with output dimension equal to the number of actions and followed by a softmax, I trained over the data for 5 epochs. Testing was done by running the model for 1,000 episodes in the CartPole environment with the reward-to-go conditioning as specified in Appendix A.

For training the CQL, offline DDQN, and BC algorithms, the implementation provided by d3rlpy ([SI22]) was used with default parameters. Each algorithm was trained on the three datasets for 10 epochs and then evaluated on 1000 episodes in the real CartPole environment.

4 Results

Figure 1 shows the results for Decision Transformer, CQL, and offline DDQN across the different datasets below.

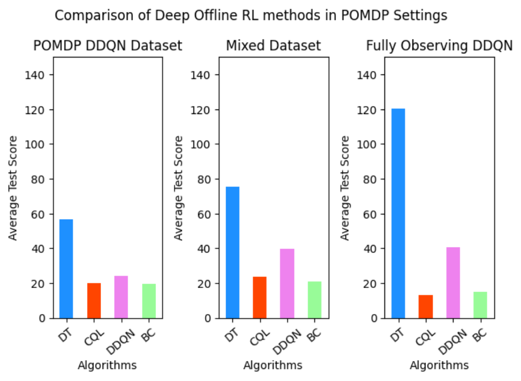

Figure 1: Comparison of Decision Transformer (DT) in blue, Conservative Q-Learning (CQL) in red, Double Deep Q-Network (DDQN) in purple, and Behaviour Cloning (BC) in green over the three different datasets for the CartPole task

Notably, no agent achieves a perfect policy of 500, this points to the inability of learning optimal policies in partially observable offline settings when there are unobserved confounders influencing the expert policy. For CQL and DDQN, these results are not suprising given that the agents are never able to observe or glean what the velocities may be from only one state. Using the language of causal inference we can realize why this is the case, an underlying assumption in RL is that the conditional distributions in the data $P(R_T|s_T, a_T), P(R_T|s_T, do(a_T))$ are equal since the observed R_T's are the result of do-actions which were chosen with respect to S_T, but, when the agent deciding which action to take chooses based on a different S_T we get $P(R_T|s_T, a_T) \neq P(R_T|s_T, do(a_T))$ since there is unobserved confounding by the velocities which govern the action choices in the dataset. The inability of the Behaviour Cloning algorithms to achieve perfect performance shows the significance of knowing the velocities for each state to properly choose actions and maximize reward. While the Decision Transformer outperforms the other algorithms it also fails to achieve perfect performance, despite the potential ability to closely approximate velocities given the recent histories, this points toward an inability to extract latent temporal factors upon which good policies rely on, such as the velocities in this case.

As for comparing the performances of the individual agents, Decision Transformer outperforms CQL, DDQN, and BC on all datasets showing that its ability to handle sequences and histories improves performance in offline POMDP tasks. As expected the algorithms generally improve as the dataset gains more entries from the fully observing DDQN agent which did attain perfect performance. DDQN outperforms both CQL and BC in all the datasets pointing towards the inability to directly learn what the expert agents are doing that makes them successful when the expert agents choose actions based off of unobserved values. The better performance from the Decision Transformer points towards Decision Transformers being able to separate what make good policies and bad policies more effectively than the other algorithms, but, its inability to learn a perfect policy from the fully observing expert dataset shows that it cannot differentiate completely.

To further understand the behaviour of the algorithms and their behaviour I simulated each algorithm, trained on the fully observing DDQN dataset, for 100 episodes with each episode starting at the same state. The starting state had everything equal to 0 except for the angle which was set at 0.01 radians. Figure 2 compares the trajectories that the different agents followed over the 100 episodes. The Decision Transformer is able to keep the pole angle close to 0 but unable to course-correct the pole once it gets far enough away from being at 0 degrees. This may point toward an inability to generalize outside of the data distribution observed since the fully observing DDQN agent learns to keep the pole upright and so there are minimal positive examples of what to do when the pole moves away from 0. DDQN performs a form of bang-bang control but due to the inability to observe the velocities of the angle oscillates with increasing amplitude until ultimately passing the episode termination threshold. Lastly, CQL and BC are unable to even perform bang-bang control and the pole instantly goes towards the thresholds point towards an inability to differentiate why the expert agents take actions in certain states that leads to confusion in the policy.

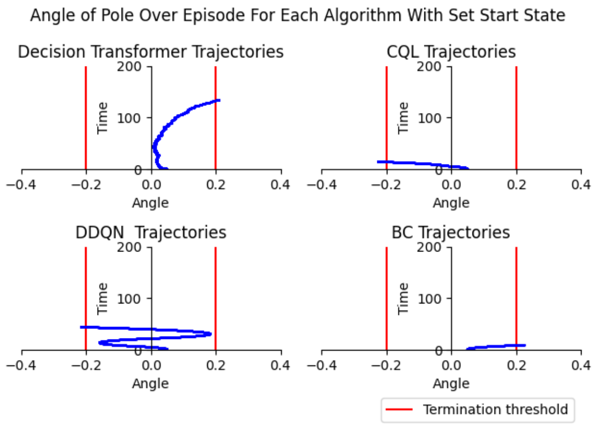

Figure 2: Plot showing the angle of the pole as time increases for each agent

5 Conclusion and Future Work

This project shows the performance of various deep offline RL agents when put in non-markovian decision processes. The results point towards the strength of Decision Transformers relative to other agents in such problem settings. However, the performance of all offline agents is poor and agents are unable to learn optimal policies even when they occur in the dataset. Additionally we see an inability in the Decision Transformer to course correct when the pole angle goes away from being upright suggesting an inability to generalize to out of distributions states.

This work still leaves many questions still unanswered about Decision Transformers in offline POMDP setting, and raises some about general deep sequence-models for offline POMDP RL. Due to the impressive performance of large language models and the observed ability of Transformers to vastly improve with scale, running this experiment with greater scale may give different results and provide better Decision Transformer agents. Another interesting research direction is the use of sequence-to-sequence models for offline POMDP tasks, here Decision Transformer was only compared to deep RL algorithms which assume the markovian assumption, a comparison to an offline LSTM agent would yield interesting insights into the general performance of sequence-to-sequence architectures in offline POMDP settings. Lastly, the Decision Transformer uses multi-head attention in a specified way that works well for language but POMDP's are a different type of sequential task and so different model architectures which incorporate attention in a way that is more consistent with RL problems may outperform the Decision Transformer, more research into such architectures would surely enhance the understanding of sequence-to-sequence modelling in RL.

References

[Vas+17] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: `1706.03762 [cs.CL]`.

[Kum+20] Aviral Kumar et al. *Conservative Q-Learning for Offline Reinforcement Learning*. 2020. arXiv: `2006.04779 [cs.LG]`.

[Che+21] Lili Chen et al. *Decision Transformer: Reinforcement Learning via Sequence Modeling*. 2021. arXiv: `2106.01345 [cs.LG]`.

[SI22] Takuma Seno and Michita Imai. "d3rlpy: An Offline Deep Reinforcement Learning Library". In: *Journal of Machine Learning Research* 23.315 (2022), pp. 1–20. URL: `http://jmlr.org/papers/v23/22-0017.html`.

6 Appendix A

Model architecture for Decision Transformer agents.

Hyperparameter	**Value**
Number of layers	4
Number of attention heads	2
Embedding dimension	32
Nonlinearity function	R
Dimension of feedforward networks	32
Batch Size	32
Context length K	16
Return-to-go conditioning	600
Dropout	0.0
Learning Rate	0.00001