Decision Transformer Reward Prediction(DTRP): Application of Transformer in Reinforcement Learning

Ying Chen*, Tianyu Liang*, Yike Wang*
University of California, Berkeley
Berkeley, CA
{ying-chen, tianyul, yike_wang}@berkeley.edu

December 15, 2022

Abstract

Recent developments in the field of sequence modeling using deep neural networks have greatly increased their effectiveness, including LSTMs, sequence-to-sequence models, and Transformer architectures with self-attention [4]. In view of these advances, researchers have started to consider how such sequence models could be used to improve the performance of RL algorithms, which are also based on sequential processes. Previously transformers are used as an architectural choice for components within traditional RL algorithms. In recent years, researchers are trying to use a Transformer architecture to model overall distributions over trajectories. Rather than fitting value functions or computing policy gradients, Janner et al. 2021 [3] used state-of-art Transformer architectures to model distributions over sequences of states, actions, and rewards. Chen et al. 2021 [1] leveraged a causally masked Transformer to output the optimal actions. The simplicity and scalability of the Transformer architecture with its competitive results, exceeding model-free and model-based offline RL baselines on various tasks, lead us to explore more on the potential benefits and limitations of using Transformers for sequential decision making in Deep Reinforcement Learning.

In this report, We explore the possibility of using transformers to learn the reward function. This is of great help in situations where we might not have easy access to rewards(sparse reward settings), so it would be desirable for the machine to learn a reward function to measure the quality of its output at arbitrary point without having to pay a high cost (i.e. have expert to label the data). Besides, we find that we could also use transformers to predict actions, which is helpful in exploration or generating training data without interaction with the environment.

We refer to our algorithm as Decision Transformer Reward Prediction (DTRP) and evaluate it over three standard continuous control benchmark tasks in Mujoco [5] including Half Cheetah, Hopper, and Walker. We found DTRP outperforms conventional RL algorithms and the original Decision Transformer on almost all tasks.

1 Reinforcement learning with Transformer

1.1 Offline Reinforcement Learning

Offline reinforcement learning is a method for training RL agents using pre-collected, fixed data sets rather than online interactions with the environment. This approach allows for the use of larger and more diverse datasets than would be possible with online learning, potentially leading to improved agent performance.

Suppose we have a trained policy $\pi_\theta(\tau)$, and we have samples from another policy $\bar{\pi}(\tau)$, we can use the samples from $\bar{\pi}(\tau)$ to calculate $J(\theta)$ function using importance sampling:

$$J(\theta) = E_{r\sim\pi_\theta(\tau)}[r(\tau)] = \int \pi_\theta(\tau) r(\tau) d\theta = \int \frac{\pi_\theta(\tau)}{\pi_\theta(\tau)} \pi_\theta(\tau) r(\tau) d\theta = E_{r\sim\pi_\theta(\tau)}[\frac{\pi_\theta(\tau)}{\bar{\pi}(\tau)} r(\tau)]$$

1.2 Decision Transformers [1]

One way that transformers can be used in deep reinforcement learning is by representing the trajectory dataset as a sequence of (state, action, reward), and then using a transformer model to process this sequence and make predictions about the actions that the agent should take for current state. With learnt embedding of state, action, reward as tokens, we could predicts future action tokens via autoregressive modeling. This can help the agent learn a more compact and generalizable representation of the history, which can be useful for making better decisions.

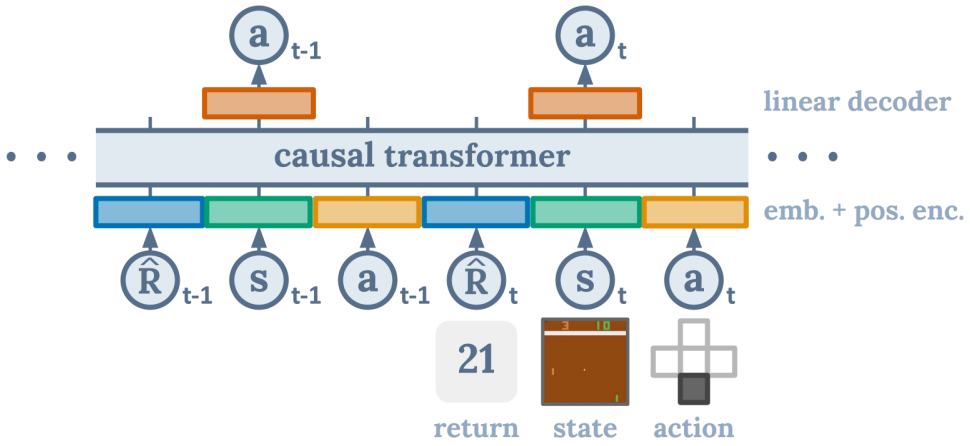

Figure 1: Decision Transformer Architecture[1]

For finding the shortest path on a reward graph, a Decision Transformer model is trained on a dataset of random walks, and the model is able to learn to stitch together subsequences from different suboptimal training trajectories in order to produce optimal trajectories at test time. This behavior is similar to that of off-policy Q-learning algorithms commonly used in offline reinforcement learning frameworks, but is achieved without the need for TD learning algorithms, value pessimism, or behavior regularization. Additionally, this approach allows for conditional generation, where we can initialize a trajectory by inputting our desired return. The result is a model that can generate a wide distribution of policies.

1.3 Proposed Modifications

- We use the transformers to predict not only actions, but also states and returns, with both previous return and state-action pair. In particular, after feeding the input embedding into the transformer, we will then use some simple linear network layer to fit the output to the target we desire. We train it so that the predicted state and return to go will depend on both the input state and action pair. Furthermore, the predicted action will not only depend on previous state, but also the previous return to go.

- When predicting the return, a ReLU activation layer is added after the linear sequential layer as we suppose all returns(reward-to-go) shall be positive.

- The original transformer RL implementation uses the reward provided by the environment to calculates the appropriate return to go. And it simply ignored the values of rewards if under sparse settings and relied on the robustness of the network to handle the training. However, we want to adapt the implementation by incorporating predicted return-to-go to see if the transformer can perform better if no actual reward is provided. This simulates situations in real world where data label is expensive and not always available.

2 Experiments

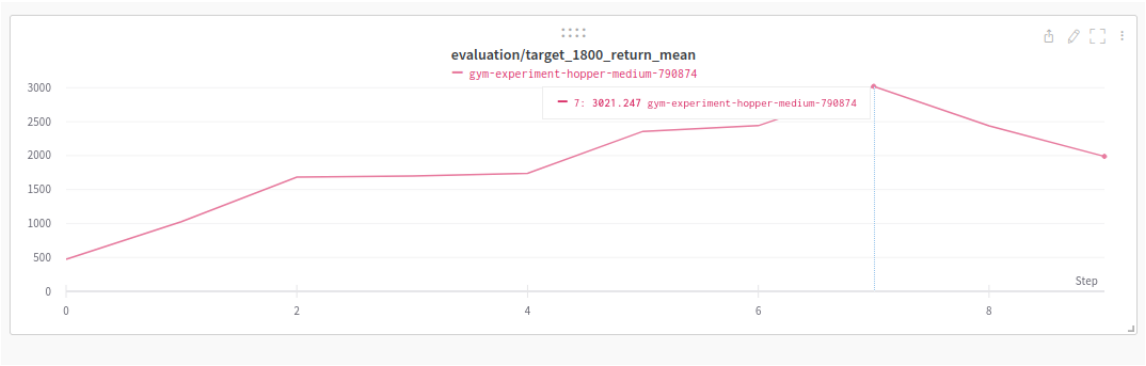

Figure 2: hopper medium with initial return-to-go set to 1800

Figure 3: hopper medium with initial return-to-go set to 3600

Figure 4: hopper medium-expert with initial return-to-go set to 3600

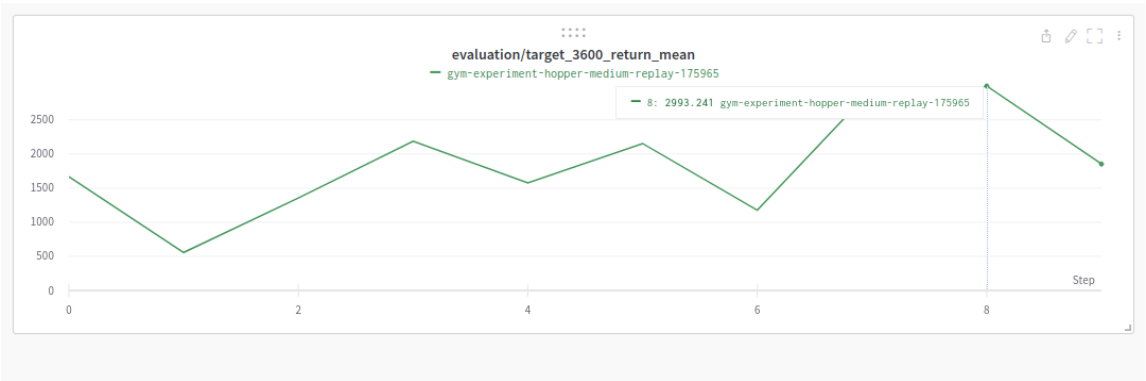

Figure 5: hopper medium-replay with initial return-to-go set to 3600

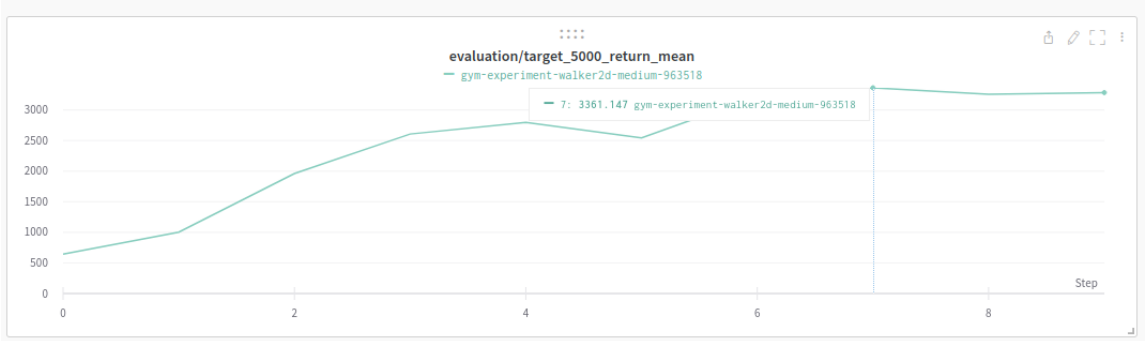

Figure 6: walker medium with initial return-to-go set to 5000

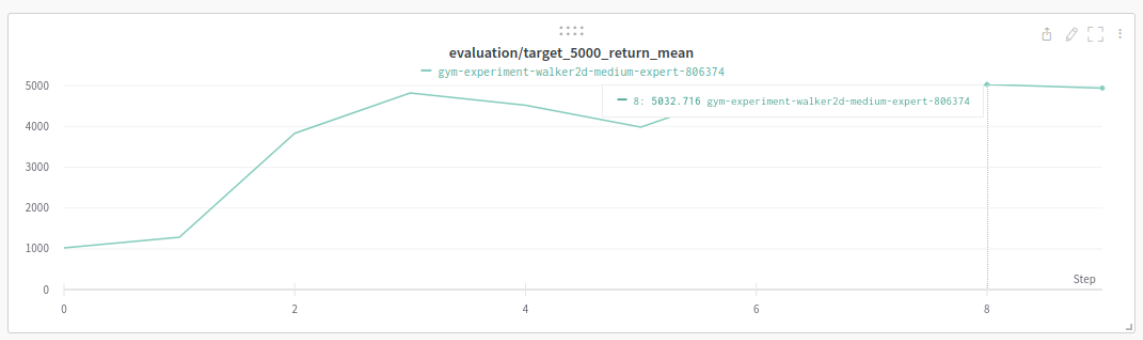

Figure 7: walker medium-expert with initial return-to-go set to 5000

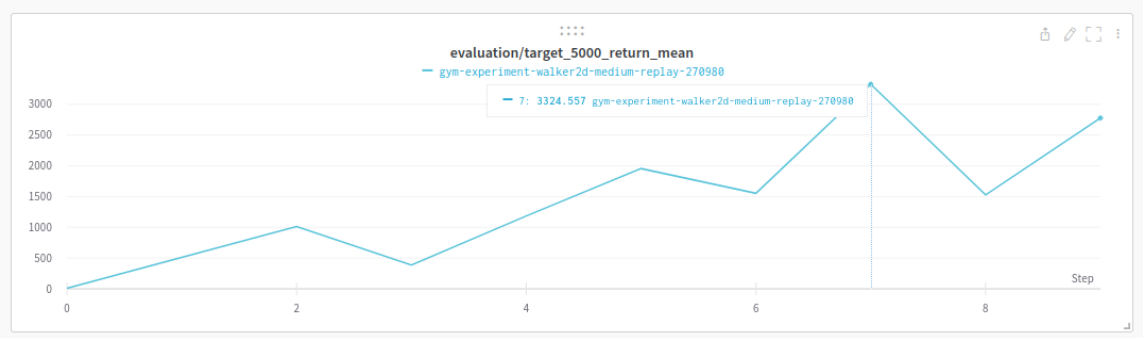

Figure 8: walker medium-replay with initial return-to-go set to 5000

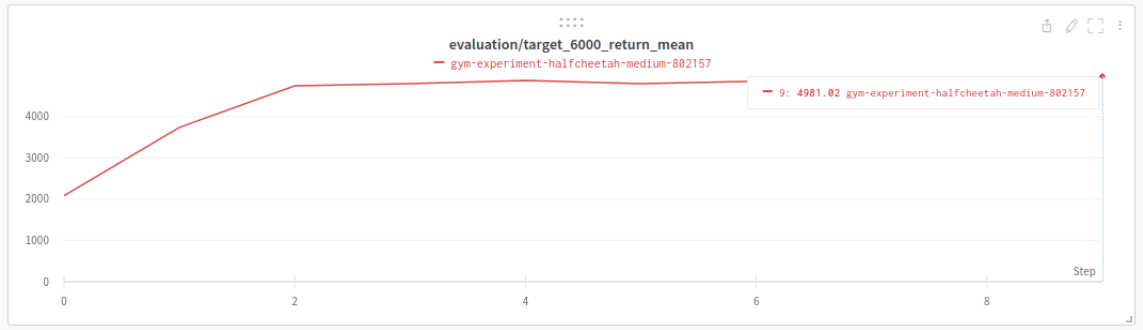

Figure 9: halfcheetah medium with initial return-to-go set to 6000

Dataset	Environment	DTRP (our)	DT(Chen)	CQL	BRAC-v
Medium-Expert	HalfCheetah	49.34	**86.8 ± 1.3**	62.4	41.9
Medium-Expert	Hopper	105.65	**107.6 ± 1.8**	98.7	0.8
Medium-Expert	Walker	109.59	108.1 ± **0.2**	**111.0**	81.6
Medium	HalfCheetah	42.42	42.6 ± 0.1	44.4	**46.3**
Medium	Hopper	**93.38**	67.6 ± 1.0	79.2	31.1
Medium	Walker	73.18	74.0 ± 1.4	58.0	**81.1**
Medium-Replay	HalfCheetah	37.48	36.6 ± 0.8	46.2	**47.7**
Medium-Replay	Hopper	**92.50**	82.7 ± 7.0	48.6	0.6
Medium-Replay	Walker	**72.38**	66.6 ± 3.0	26.7	0.9
Average (All Settings)	**69.2**	54.2	–	–	47.7

Table 1: Results for D4RL datasets [2]. We report the mean and variance for three seeds. DTRP outperforms conventional RL algorithms and the original Decision Transformer on almost all tasks.

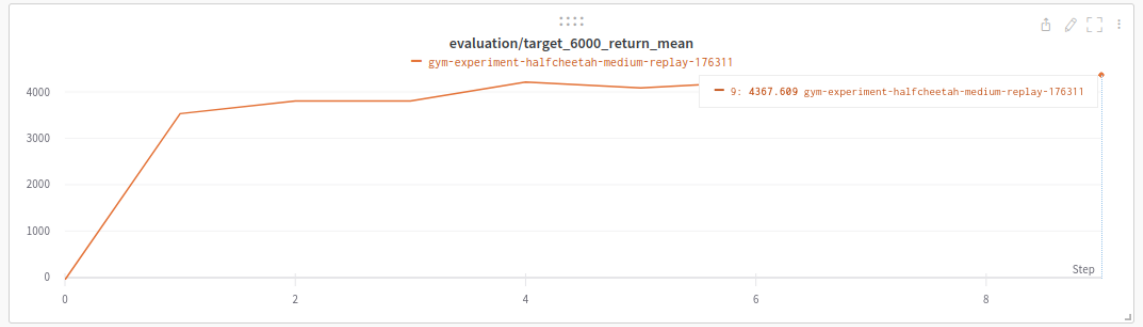

Figure 10: halfcheetah medium-replay with initial return-to-go set to 6000

3 Discussion

Overall, our results demonstrated improvements in quite a few settings, showing that transformer can indeed be adapted to learning rewards for sparse settings. However, we notice that for the halfcheetah medium-expert setting, the performance is extremely poor compared to the original implementation. This is caused by the dependency in our implementations; we designed it so that the output action will depend on both the previous inputs state and return-to-go. This approach worked well in all other settings except this one. This is likely due to the relatively higher number of low reward trajectories that are contained in the training dataset, which

can interfere with the quality of return-to-go.

In reinforcement learning, return-to-go of previous states can help us to learn better about action prediction because for highly-correlated (state, action, reward) sequences, it allows the learning algorithm to take into account the potential future rewards that may result from taking a particular action in a given state. By considering the potential future rewards, the learning algorithm is able to better predict which actions are likely to lead to the greatest overall reward, and can therefore learn to take these actions more frequently in order to maximize the rewards it receives. Additionally, using return-to-goof previous states can help to reduce the impact of noisy or unreliable rewards, allowing the learning algorithm to focus on the long-term rewards that are likely to result from taking a particular action.

We also tried other approaches such as planning based methods. However, this approach is infeasible because the randomly generated actions are simply inferior compared to the ones outputted by the transformer model. Furthermore, it is also computationally inefficient due to having to evaluate extra trajectories.

References

[1] Lili Chen et al. *Decision Transformer: Reinforcement Learning via Sequence Modeling*. 2021. DOI: 10.48550/ARXIV.2106.01345. URL: https://arxiv.org/abs/2106.01345.

[2] Justin Fu et al. *D4RL: Datasets for Deep Data-Driven Reinforcement Learning*. 2020. DOI: 10.48550/ARXIV.2004.07219. URL: https://arxiv.org/abs/2004.07219.

[3] Michael Janner, Qiyang Li, and Sergey Levine. *Offline Reinforcement Learning as One Big Sequence Modeling Problem*. 2021. DOI: 10.48550/ARXIV.2106.02039. URL: https://arxiv.org/abs/2106.02039.

[4] Ashish Vaswani et al. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762.

[5] Duan Yan et al. *Benchmarking Deep Reinforcement Learning for Continuous Control*. 2016. DOI: 10.48550/ARXIV.1604.06778. URL: https://arxiv.org/abs/1604.06778.

4 Contribution

Ying: Literature review, concluding experiments, paper writing
Tianyu: code implementation, benchmark experiments
Yike: literature review, study design, paper writing