Towards Better Offline RL with Structured State Space Sequence (S4) Models

Kaustubh Dighe* **Muhender Raj Rajvee*** **Shayan Pardis***
{kdighe, muhender, shayanp}@mit.edu
Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

1 Draft

- at first we thought training GPT would be challenging but we overcame this challenge by looking for minimalistic implementation of GPT models. We explored minGPT and nanoGPT. We also found reference notebook?. The gpt model that we used was very minimalistic and had 3 blocks, 1 attention head, and an embedding dimensionality of 128. This had roughly 1 million parameters and we were able to run and train it on a laptop.

  we should put the repo here as well

Discussion and comparison between S4 and DT:

- a large context length makes it extremely slow... as the size of attention matrix should is $maxt \times maxt$, but S4 doesn't have this limitation. Just for comparison, we used contextlength=32 but in the door key the episode length is 100 verify this? on average!

2 Introduction

Janner, Li, and Levine[1] introduced the idea of viewing reinforcement learning as a sequence generation problem. This allows utilizing the machinery developed for large-scale unsupervised learning. Introduced in Chen *et al.*[2], Decision Transformers (DT) is one of the most successful realization of this idea. Among other metrics, the paper evaluates the model with credit assignment as a metric.

Credit assignment is a fundamental problem in reinforcement learning, that of measuring the influence of an action on future rewards. Figuring out which actions contributed the most to reach the goal or maximum reward distinguishes luck from skill. Credit assignment is therefore a very useful metric for evaluating sparse-reward environments. It requires capturing long-term relationships between actions and rewards. Capturing long-term dependence is also the goal in sequence modeling.

On the sequence models front, there has been work on another class of sequence models - Structured State Space Sequence (S4). S4 models sequences with differential equations and is asymptotically faster than transformers, allowing for much larger sequences.

In this project, we study the impact of using S4 models in place of transformers in DT in various carefully selected different sets of environments. We find that the S4 sequence models perform better in all environments that we tested. They are also several times faster than DT. Additionally, since S4 models the sequences as a differential equations, they show a much more improvement in physical environments like cart pole which have a continuous state space.

*equal contribution

3 Background and Relevant Literature

Conditioning on Reward

The idea of conditioning the policy on future return is critical for credit assignment problem (e.g. $\pi(a|s, R)$ where R is often called return to go). In a sense, this combines the value function and policy together and allows the algorithm to learn from trajectories generated from other policies. Policy Gradients Incorporating the Future (Venuto *et al.*[3]) proposes a method that uses information from future states to improve policy gradients in the present, but does not try to solve the credit assignment problem directly

Structured State Space Models (S4)

S4 ([4]) models the sequence as a differential equation in a latent space and thus is able to capture long range dependencies. Unlike other RNN models where sequentially reading the data is the time bottleneck, the encoding process in S4 can be parallelized using Fourier Transforms. Also, unlike transformers, S4 does not need to keep all the history at inference. Thus S4 showcases better performance in both training and inference.

While we focus on the credit assignment metric which usually applies to environment with discrete tasks, Lu *et al.*[5] are interested in tasks in the control domain, which are non-discrete. S4 naturally lends itself to modeling the dynamics of these systems, the results of which can be seen in their work. We on the other hand try applying S4 models to discrete tasks to look for similar improvements with respect to the credit assignment metric.

Decision Transformers

Since RL can be framed as a sequence modeling task, we can use sequence models like transformers to solve it. As introduced by Chen *et al.*[2], DT can be used to solve offline RL tasks by passing in the trajectory-wise states, actions, and rewards as separate tokens to the model.

Credit Assignment Problem

Previous work like [6] discusses the limitations of existing Hindsight Credit Assignment (HCA) algorithms in deep RL that lead to their poor performance or complete lack of training, then propose several theoretically-justified modifications to overcome them. [3] and decision transformers are ways to do credit assignment better than HCA.

4 Methods

We have tried running the official decision transformers repository (github.com/kzl/decision-transformer) and some other reference implementations from HuggingFace. However, each of them have issues with package compatibility and more importantly they do not have support for easily replacing sequence models. Hence, we decided to implement our own decision transformer notebook based on the official implementation. For GPT training, access V100 and A100 GPUs on Colab Pro. Also, from the official decision transformers repository, we came across minGPT [7] which is a smaller and easier version of OpenAI's GPT. We also plan to experiment with that and see if it works better/faster as a concern in this project is long training times. Since decision transformers are an offline reinforcement learning technique, we need to generate a lot of trajectories. We will experiment with randomly generating trajectories or having a PPO model generate a better ensemble of trajectories.

GPT training time wasn't as much of an issue as we originally thought because we used <fill in>.

have a table to how the speed of each of the tasks... As

- As the first step, we tried reproduce the experiments in Decision Transformer [2] paper

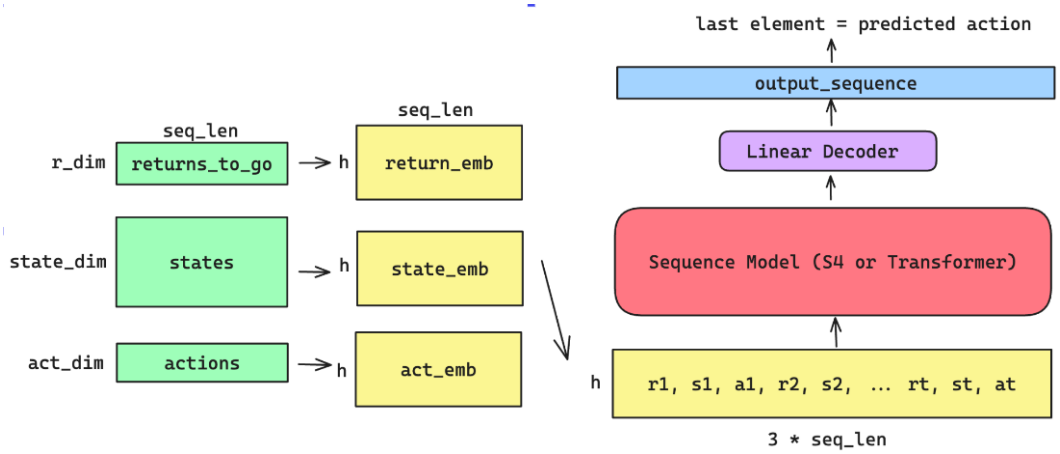

Figure 1: Decision Sequence models include passing the r, s, a for each step in the trajectory

5 Experiments

As we have mentioned in passing several times by now, we have a lot of factors to vary and we plan to experiment them. We have picked a variety of environments to benchmark performance. Each of the environments is carefully chosen to measure performance of algorithms from different aspects. In addition to that, we will vary the sequence model inside the decision transformer.

Another confusion is on how to view the trajectory as a sequence. The way the decision transformer paper does it is to send state, action and return as 3 separate tokens one by one. It works well, however we have a concern that the transformer needs to spend additional effort on learning this relationship between adjacent triplets of tokens. We want to see if sending all 3 concatenated into one token works well or not. The objective function as prescribed by [2] is just the mean loss between the predicted action tokens by the DT and the actions in our trajectory. The loss can be L2 for continuous actions and cross entropy for discrete.

Random Walk Environment

In this environment the goal is reaching the goal node in a graph G starting from a node $s \sim D_0$. The reward is the negative length of the path traversed, so essentially the agent needs to find the shortest path in this graph given some random walks. The state space would be all vertices in the graph and actions would be traversing any outgoing edge from the vertex the agent is currently at. The objective function is the cross entropy loss between the predicted actions by the DT and the actions in our trajectory.

The challenge in this environment is that agent needs to stitch paths that are traversed in dataset and find a path that is better than the shortest path in the dataset. In [2] author suggested that DT is able to achieve this.

The Umbrella Length

This task, which is a part of the Behavior Suite for RL (bsuite) [8], is also specifically for testing credit assignment. As described in Venuto *et al.*[3], The task involves a long sequential episode where only the first observation (a forecast of rain or shine) and action (whether to take an umbrella) matter, while the rest of the sequence contains random unrelated information with random rewards except the last state. The state conveys information about having an umbrella and the forecast. More specifically, it has 3 pieces of information: need_umbrella, have_umbrella, and time_to_live, and n distractors that are randomly sampled from a binomial distribution. The action space is whether to take an umbrella or not. A reward of +1 is given at the end of the episode if the agent chooses correctly whether to take the umbrella or not depending on the forecast, and -1 otherwise. The "length" part comes from the amount of distracting information in the sequence in the form of the

chain length, which is the number of states until the final state. The objective function is the cross entropy loss between the predicted actions by the DT or S4 policy and the actions in our trajectory.

Door Key Environment

This minigrid environment was specifically designed as a testbed where credit assignment is hard and is necessary for success (Hung *et al.*[9]). In this environment, the agent has to pick up a key (which has no immediate reward) and open a door to a room that has 10 apples and the agent gets rewarded by picking those. In addition to sparse rewards, learning to associate the credit to the actions necessary for picking up the key makes this environment challenging. Additionally, there can also be a third room that the agent can enter after finding its key and the agent will get a small reward after entering the room. Learning to go from the second room is specifically challenging since the small reward of third room will be faded away by the high reward of picking apples in the second room.

This environment is also discrete. The actions are moving left, right, up, down, picking up, going to the next room. The states are the grid coordinates coupled with the room we are in. The reward is 10 if we pick up the 10 apples in the 2nd room and and additional 1 if we reach the 3rd room. The reward is 0 otherwise. The objective function is the cross entropy loss between the predicted actions by the DT and the actions in our trajectory.

Cartpole Environment

In this environment, a cart has a vertical pole hinged on it. The goal is to keep the pole from falling down. Actions are discrete - pushing the cart forward or backward. The reward is +1 for each time step that the pole does not fall. When the pole falls or we are above 500 steps, the episode ends. The observation space is continuous, as shown in **??**.

6 Contribution

References

[1] M. Janner, Q. Li, and S. Levine, *Offline reinforcement learning as one big sequence modeling problem*, 2021. arXiv: 2106.02039 [cs.LG].

[2] L. Chen, K. Lu, A. Rajeswaran, *et al.*, "Decision transformer: Reinforcement learning via sequence modeling," *CoRR*, vol. abs/2106.01345, 2021. arXiv: 2106.01345. [Online]. Available: https://arxiv.org/abs/2106.01345.

[3] D. Venuto, E. Lau, D. Precup, and O. Nachum, "Policy gradients incorporating the future," *CoRR*, vol. abs/2108.02096, 2021. arXiv: 2108.02096. [Online]. Available: https://arxiv.org/abs/2108.02096.

[4] A. Gu, K. Goel, and C. Ré, *Efficiently modeling long sequences with structured state spaces*, 2022. arXiv: 2111.00396 [cs.LG].

[5] C. Lu, Y. Schroecker, A. Gu, *et al.*, *Structured state space models for in-context reinforcement learning*, 2023. arXiv: 2303.03982 [cs.LG].

[6] V. Alipov, R. Simmons-Edler, N. Putintsev, P. Kalinin, and D. Vetrov, *Towards practical credit assignment for deep reinforcement learning*, 2022. arXiv: 2106.04499 [cs.LG].

[7] A. karpathy, *Mingpt: A minimal pytorch re-implementation of the openai gpt training*, 2020. [Online]. Available: https://github.com/karpathy/minGPT.

[8] I. Osband, Y. Doron, M. Hessel, *et al.*, "Behaviour suite for reinforcement learning," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rygf-kSYwH.

[9] C. Hung, T. P. Lillicrap, J. Abramson, *et al.*, "Optimizing agent behavior over long time scales by transporting value," *CoRR*, vol. abs/1810.06721, 2018. arXiv: 1810.06721. [Online]. Available: http://arxiv.org/abs/1810.06721.