Decision Transformer : Reinforcement Learning via Sequence Modeling

April 15, 2025

Indian Institute of Science, Bangalore

Team: Neural Knots

Ashwin K.M. (23882), Kuldeep Jatav (23684), Udit Shah (23475)

Table of Contents

Research Problem and Goal

Research Problem

- Can reinforcement learning be reframed as a sequence modeling problem using transformers?

Research Problem and Goal

Research Problem

- Can reinforcement learning be reframed as a sequence modeling problem using transformers?
- Can we bypass value functions and policy gradients by predicting actions through conditional sequence modeling?

Research Problem and Goal

Research Problem

- Can reinforcement learning be reframed as a sequence modeling problem using transformers?
- Can we bypass value functions and policy gradients by predicting actions through conditional sequence modeling?
- How effectively can this be applied to offline RL using fixed datasets?

Goal

- Adapt the Decision Transformer (DT) to work with Minari datasets and the MuJoCo control suit.

Research Problem and Goal

Research Problem

- Can reinforcement learning be reframed as a sequence modeling problem using transformers?
- Can we bypass value functions and policy gradients by predicting actions through conditional sequence modeling?
- How effectively can this be applied to offline RL using fixed datasets?

Goal

- Adapt the Decision Transformer (DT) to work with Minari datasets and the MuJoCo control suit.
- Evaluate DT performance across multiple control tasks and compare with Behavior Cloning (BC).

Research Problem and Goal

Research Problem

- Can reinforcement learning be reframed as a sequence modeling problem using transformers?
- Can we bypass value functions and policy gradients by predicting actions through conditional sequence modeling?
- How effectively can this be applied to offline RL using fixed datasets?

Goal

- Adapt the Decision Transformer (DT) to work with Minari datasets and the MuJoCo control suit.
- Evaluate DT performance across multiple control tasks and compare with Behavior Cloning (BC).
- Simulate benchmark environments to validate DT functionality.

Why Decision Transformer?

Challenges in Offline RL

- **Distributional shift:** Policies may query out-of-distribution states not seen during training.

Why Decision Transformer?

Challenges in Offline RL

- **Distributional shift:** Policies may query out-of-distribution states not seen during training.
- **Compounding errors:** Inaccurate actions snowball over time, destabilizing learning.

Why Decision Transformer?

Challenges in Offline RL

- **Distributional shift:** Policies may query out-of-distribution states not seen during training.
- **Compounding errors:** Inaccurate actions snowball over time, destabilizing learning.
- **Value overestimation:** Learning value functions offline often leads to inflated estimates.

Advantages of Supervised Learning with Transformers

- **Stable credit assignment:** Self-attention layers can directly associate rewards with prior states.

Why Decision Transformer?

Challenges in Offline RL

- **Distributional shift:** Policies may query out-of-distribution states not seen during training.
- **Compounding errors:** Inaccurate actions snowball over time, destabilizing learning.
- **Value overestimation:** Learning value functions offline often leads to inflated estimates.

Advantages of Supervised Learning with Transformers

- **Stable credit assignment:** Self-attention layers can directly associate rewards with prior states.
- **No bootstrapping or discounting:** Avoids the instability from temporal difference learning.

Why Decision Transformer?

Challenges in Offline RL

- **Distributional shift:** Policies may query out-of-distribution states not seen during training.
- **Compounding errors:** Inaccurate actions snowball over time, destabilizing learning.
- **Value overestimation:** Learning value functions offline often leads to inflated estimates.

Advantages of Supervised Learning with Transformers

- **Stable credit assignment:** Self-attention layers can directly associate rewards with prior states.
- **No bootstrapping or discounting:** Avoids the instability from temporal difference learning.
- **Scalability and generalization:** Leverages large-scale pretraining techniques from language models.

What is DT? – Sequence Modeling Perspective

Trajectory as a Sequence

- Instead of learning value functions or policies, DT models trajectories as sequences.

What is DT? – Sequence Modeling Perspective

Trajectory as a Sequence

- Instead of learning value functions or policies, DT models trajectories as sequences.
- Inputs are organized as:

$$(\hat{R}_1, s_1, a_1, \hat{R}_2, s_2, a_2, \ldots, \hat{R}_T, s_T, a_T)$$

where $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$ is return-to-go.

What is DT? – Sequence Modeling Perspective

Trajectory as a Sequence

- Instead of learning value functions or policies, DT models trajectories as sequences.
- Inputs are organized as:
  $$(\hat{R}_1, s_1, a_1, \hat{R}_2, s_2, a_2, \dots, \hat{R}_T, s_T, a_T)$$
  where $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$ is return-to-go.
- At test time, DT is conditioned on desired return and current state to generate actions autoregressively.

What is DT? – Architecture and Training

Architecture

- Input: Last K timesteps $\rightarrow 3K$ tokens (return, state, action).

What is DT? – Architecture and Training

Architecture

- Input: Last K timesteps $\rightarrow 3K$ tokens (return, state, action).
- Each modality is projected via a separate linear layer; visual inputs use CNN encoders.

What is DT? – Architecture and Training

Architecture

- Input: Last K timesteps $\rightarrow 3K$ tokens (return, state, action).
- Each modality is projected via a separate linear layer; visual inputs use CNN encoders.
- Learned timestep embeddings added to each token (1 timestep = 3 tokens).

What is DT? – Architecture and Training

Architecture

- Input: Last K timesteps $\rightarrow 3K$ tokens (return, state, action).
- Each modality is projected via a separate linear layer; visual inputs use CNN encoders.
- Learned timestep embeddings added to each token (1 timestep = 3 tokens).
- Uses a GPT-like transformer with causal attention to predict future actions.

Training

- Trained on offline trajectories using supervised learning.

What is DT? – Architecture and Training

Architecture

- Input: Last K timesteps $\rightarrow 3K$ tokens (return, state, action).
- Each modality is projected via a separate linear layer; visual inputs use CNN encoders.
- Learned timestep embeddings added to each token (1 timestep $=$ 3 tokens).
- Uses a GPT-like transformer with causal attention to predict future actions.

Training

- Trained on offline trajectories using supervised learning.
- Objective: predict a_t given $(\hat{R}_t, s_t)$ using cross-entropy or MSE loss.

Experiments Review

- Decision Transformer(DT)
- Behavioural Cloning (BC)

Experiments Review

- Decision Transformer(DT)
- Behavioural Cloning (BC)

for each DT and BC we have conducted 11 Experiments as follows

Environment	Simple	Medium	Expert
Half-Cheetah	✓	✓	✓
Hopper	✓	✓	✓
Walker-2D	✓	✓	✓
Reacher	-	✓	✓

Table: Experiments Table

* We didn't conducted reacher simple because dataset doesn't exist for it.
* We have recorded videos for each experiment.

Comparison

Dataset	Environment	DT (Ours)	10% BC	25% BC	40% BC	100% BC
Simple	HalfCheetah	40.90	40.8	45.60	43.1	40.2
Simple	Hopper	75.23	72.57	71.2	70.8	66.9
Simple	Walker	82.21	84.10	80.9	78.8	77.3
Medium	HalfCheetah	44.01	45.31	46.1	45.1	44.09
Medium	Hopper	92.03	87.4	86.13	78.3	77.3
Medium	Walker	73.99	75.6	70.21	66.2	41.3
Medium	Reacher	31.7	35.0	36.9	37.2	44.2
Expert	HalfCheetah	97.02	101.3	97.0	97.5	98.0
Expert	Hopper	119.7	114.2	112.5	109.7	106.1
Expert	Walker	120.3	129.4	121.7	117.2	94.8
Expert	Reacher	65.0	66.0	67.0	68.0	69.0

Table 1: Score Table for Environments and Datasetsr

These scores represent the normalized score where 100 represents the score of expert policy.

Atari - Breakout

- Attempted to implement DT on Atari `DQN Replay Buffer` data.

Atari - Breakout

- Attempted to implement DT on Atari `DQN Replay Buffer` data.
- Original dataset unavailable (deprecated/private on Google Cloud).

Atari - Breakout

- Attempted to implement DT on Atari `DQN Replay Buffer` data.
- Original dataset unavailable (deprecated/private on Google Cloud).
- Used alternative expert policy dataset from `Minari`.

Atari - Breakout

- Attempted to implement DT on Atari `DQN Replay Buffer` data.
- Original dataset unavailable (deprecated/private on Google Cloud).
- Used alternative expert policy dataset from `Minari`.
- But this data limited to only 10 trajectories per game.

Atari - Breakout

- Attempted to implement DT on Atari `DQN Replay Buffer` data.
- Original dataset unavailable (deprecated/private on Google Cloud).
- Used alternative expert policy dataset from `Minari`.
- But this data limited to only 10 trajectories per game.
- Results showed significant performance gap:

Atari - Breakout

- Attempted to implement DT on Atari `DQN Replay Buffer` data.
- Original dataset unavailable (deprecated/private on Google Cloud).
- Used alternative expert policy dataset from `Minari`.
- But this data limited to only 10 trajectories per game.
- Results showed significant performance gap:
  - Training loss decreased over epochs.

Atari - Breakout

- Attempted to implement DT on Atari `DQN Replay Buffer` data.
- Original dataset unavailable (deprecated/private on Google Cloud).
- Used alternative expert policy dataset from `Minari`.
- But this data limited to only 10 trajectories per game.
- Results showed significant performance gap:
    - Training loss decreased over epochs.
    - But rewards were extremely low compared to original dataset.

Atari - Breakout

- Attempted to implement DT on Atari `DQN Replay Buffer` data.
- Original dataset unavailable (deprecated/private on Google Cloud).
- Used alternative expert policy dataset from `Minari`.
- But this data limited to only 10 trajectories per game.
- Results showed significant performance gap:
    - Training loss decreased over epochs.
    - But rewards were extremely low compared to original dataset.
    - Original agent: 400 rewards per iteration.

Atari - Breakout

- Attempted to implement DT on Atari `DQN Replay Buffer` data.
- Original dataset unavailable (deprecated/private on Google Cloud).
- Used alternative expert policy dataset from `Minari`.
- But this data limited to only 10 trajectories per game.
- Results showed significant performance gap:
    - Training loss decreased over epochs.
    - But rewards were extremely low compared to original dataset.
    - Original agent: 400 rewards per iteration.
    - Our implementation: much lower rewards.

Atari - Breakout

- Attempted to implement DT on Atari `DQN Replay Buffer` data.
- Original dataset unavailable (deprecated/private on Google Cloud).
- Used alternative expert policy dataset from `Minari`.
- But this data limited to only 10 trajectories per game.
- Results showed significant performance gap:
  - Training loss decreased over epochs.
  - But rewards were extremely low compared to original dataset.
  - Original agent: 400 rewards per iteration.
  - Our implementation: much lower rewards.
- **Conclusion:** Decision Transformer performance is highly sensitive to the number of sampes in the initial dataset.

Progress Towards Our Goal

Dataset Compatibility Challenges

- Adapted Decision Transformer to work with Minari datasets, which were trained using different agents.

Progress Towards Our Goal

Dataset Compatibility Challenges

- Adapted Decision Transformer to work with Minari datasets, which were trained using different agents.
- Rewrote large portions of the codebase to handle updated data structures and API changes in Minari.

Progress Towards Our Goal

Dataset Compatibility Challenges

- Adapted Decision Transformer to work with Minari datasets, which were trained using different agents.
- Rewrote large portions of the codebase to handle updated data structures and API changes in Minari.
- Dealt with deprecated environments like `gym` and `d4rl`, requiring major structural modifications.

Evaluation Framework

- Minari lacked standard benchmarks; we selected Behavior Cloning (BC) as a consistent baseline for comparison.

Progress Towards Our Goal

Dataset Compatibility Challenges

- Adapted Decision Transformer to work with Minari datasets, which were trained using different agents.
- Rewrote large portions of the codebase to handle updated data structures and API changes in Minari.
- Dealt with deprecated environments like `gym` and `d4rl`, requiring major structural modifications.

Evaluation Framework

- Minari lacked standard benchmarks; we selected Behavior Cloning (BC) as a consistent baseline for comparison.
- In the absence of automated evaluation tools, we developed a custom rendering pipeline for MuJoCo environments.

Progress Towards Our Goal

Dataset Compatibility Challenges

- Adapted Decision Transformer to work with Minari datasets, which were trained using different agents.
- Rewrote large portions of the codebase to handle updated data structures and API changes in Minari.
- Dealt with deprecated environments like `gym` and `d4rl`, requiring major structural modifications.

Evaluation Framework

- Minari lacked standard benchmarks; we selected Behavior Cloning (BC) as a consistent baseline for comparison.
- In the absence of automated evaluation tools, we developed a custom rendering pipeline for MuJoCo environments.
- This enabled validation of Decision Transformer's performance across tasks.

Contribution

- Implemented the Decision Transformer architecture.
- Designed and developed the data preprocessing pipeline.
- Contributed to the report writing, result analysis and presentation.
- Conducted experiments on the Hopper, Walker-2D, Half-Cheetah for Simple, Medium, Expert datasets & Reacher for Medium and Expert.
- Created visualizations and prepared the video demonstration.
- Worked extensively on Atari Games.
- Extensive research on theory of Decision Transformers.

***Equal Contribution**

Thank You!

Questions or Discussions Welcome