Decision Transformers

Sanjiban Choudhury
Reinforcement Learning is Hard …
Many horror stories of RL!

Nightmares of Policy Optimization
The deadly triad

- The risk of divergence arises whenever we combine three things:

  1. Function approximation
     significantly generalizing from large numbers of examples

  2. Bootstrapping
     learning value estimates from other value estimates, as in dynamic programming and temporal-difference learning

  3. Off-policy learning (Why is dynamic programming off-policy?)
     learning about a policy from data not due to that policy, as in Q-learning, where we learn about the greedy policy from data with a necessarily more exploratory policy

- Any two without the third is ok
Need many tricks to make RL work in practice!

Rainbow: Combining Improvements in Deep Reinforcement Learning

- Matteo Hessel (DeepMind)
- Joseph Modayil (DeepMind)
- Hado van Hasselt (DeepMind)
- Tom Schaul (DeepMind)
- Georg Ostrovski (DeepMind)
- Will Dabney (DeepMind)
- Dan Horgan (DeepMind)
- Bilal Piot (DeepMind)
- Mohammad Azar (DeepMind)
- David Silver (DeepMind)

Double Q Learning
Prioritized Replay
Dueling Networks
Multi-step Learning
Distributional RL
Noisy Nets

<table>
<thead>
<tr>
<th>DQN</th>
<th>no double</th>
</tr>
</thead>
<tbody>
<tr>
<td>no priority</td>
<td>no multi-step</td>
</tr>
<tr>
<td>no dueling</td>
<td>no distribution</td>
</tr>
<tr>
<td>no noisy</td>
<td>Rainbow</td>
</tr>
</tbody>
</table>

0 50 100 150 200
Millions of frames
Is there any hope?
#1 Get Data

#2 Train Policy

\[ \pi : S \rightarrow a \]

#3 Deploy!
Supervised Learning success stories
IDEA:
Can we make Reinforcement Learning look like Supervised Learning?
RvS: WHAT IS ESSENTIAL FOR OFFLINE RL VIA SUPERVISED LEARNING?

Scott Emmons\textsuperscript{1}, Benjamin Eysenbach\textsuperscript{2}, Ilya Kostrikov\textsuperscript{1}, Sergey Levine\textsuperscript{1}

\textsuperscript{1}UC Berkeley, \textsuperscript{2}Carnegie Mellon University

emmons@berkeley.edu
The Idea

- States: $s_1$, $s_2$, $s_3$
- Actions: $a_1$, $a_2$, $a_3$
- Outcomes: $\omega_1$, $\omega_2$, $\omega_3$

(a) replay buffer
The Idea

(b) training dataset
The Idea

(c) network architecture
The Algorithm

For all trajectories: \[ \max_{\theta} \sum_{\tau \in \mathcal{D}} \sum_{1 \leq t \leq |\tau|} \mathbb{E}_{\omega \sim f(\omega | \tau_{t:H})} \left[ \log \pi_{\theta}(a_t | s_t, \omega) \right]. \]

Algorithm 1 RvS-Learning

1: **Input:** Dataset of trajectories, \( \mathcal{D} = \{\tau\} \)
2: Initialize policy \( \pi_{\theta}(a | s, \omega) \).
3: **while** not converged **do**
   4: Randomly sample trajectories: \( \tau \sim \mathcal{D} \).
   5: Sample time index for each trajectory, \( t \sim [1, H] \), and sample a corresponding outcome: \( \omega \sim f(\omega | \tau_{t:H}) \).
   6: Compute loss: \( \mathcal{L}(\theta) \leftarrow \sum_{(s_t, a_t, \omega)} \log \pi_{\theta}(a_t | s_t, \omega) \)
   7: Update policy parameters: \( \theta \leftarrow \theta + \eta \nabla_{\theta} \mathcal{L}(\theta) \)
8: **end while**
9: **return** Conditional policy \( \pi_{\theta}(a | s, \omega) \)
What are some choices for “outcomes”?

Option 1: What is the future state the agent ended up at?

RvS-G (Goal conditioned)

Option 2: What is the total return that the agent got?

RvS-R (Return conditioned)
A very popular idea

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling

Felipe Codevilla, Matthias Muller, Antonio Lopez, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning


Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem

Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies


Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaskowski, and Jurgen Schmidhuber. Training agents using upside-down reinforcement learning
Do I really need to condition?
Consider the following MDP

\begin{align*}
18 + 2 + 3 & + 2 \\
-3 & -5 - 7
\end{align*}
Option 1: Return Conditioned Policy

\[ \pi(a \mid s, R) \]
Option 2: Train a policy on top returns!

\[ \pi(a \mid s) \]
An embarrassingly simply algorithm: BC%

1. Collect offline dataset using whatever behavior policy

2. Get the top % trajectories based on returns

3. Do BC on just that!
Does this even work ?!?  

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Environment</th>
<th>10%BC</th>
<th>25%BC</th>
<th>40%BC</th>
<th>100%BC</th>
<th>CQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium</td>
<td>HalfCheetah</td>
<td>42.9</td>
<td>43.0</td>
<td>43.1</td>
<td>43.1</td>
<td>44.4</td>
</tr>
<tr>
<td>Medium</td>
<td>Hopper</td>
<td>65.9</td>
<td>65.2</td>
<td>65.3</td>
<td>63.9</td>
<td>58.0</td>
</tr>
<tr>
<td>Medium</td>
<td>Walker</td>
<td>78.8</td>
<td>80.9</td>
<td>78.8</td>
<td>77.3</td>
<td>79.2</td>
</tr>
<tr>
<td>Medium</td>
<td>Reacher</td>
<td>51.0</td>
<td>48.9</td>
<td>58.2</td>
<td>58.4</td>
<td>26.0</td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>HalfCheetah</td>
<td>40.8</td>
<td>40.9</td>
<td>41.1</td>
<td>4.3</td>
<td>46.2</td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>Hopper</td>
<td>70.6</td>
<td>58.6</td>
<td>31.0</td>
<td>27.6</td>
<td>48.6</td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>Walker</td>
<td>70.4</td>
<td>67.8</td>
<td>67.2</td>
<td>36.9</td>
<td>26.7</td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>Reacher</td>
<td>33.1</td>
<td>16.2</td>
<td>10.7</td>
<td>5.4</td>
<td>19.0</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>56.7</td>
<td>52.7</td>
<td>49.4</td>
<td>39.5</td>
<td>43.5</td>
</tr>
</tbody>
</table>
Can we make this a bit more fancier?

1. Handle noisy returns

2. Collect data on-policy
### Algorithm 1 Advantage-Weighted Regression

1. $\pi_1 \leftarrow$ random policy
2. $\mathcal{D} \leftarrow \emptyset$
3. **for** iteration $k = 1, \ldots, k_{\text{max}}$ **do**
   4. add trajectories $\{\tau_i\}$ sampled via $\pi_k$ to $\mathcal{D}$
   5. $V_k^{\mathcal{D}} \leftarrow \arg \min_V \mathbb{E}_{s,a \sim \mathcal{D}} \left[ \left\| R_s^{\mathcal{D},a} - V(s) \right\|^2 \right]$
   6. $\pi_{k+1} \leftarrow \arg \max_\pi \mathbb{E}_{s,a \sim \mathcal{D}} \left[ \log \pi(a|s) \exp \left( \frac{1}{\beta} \left( R_s^{\mathcal{D},a} - V_k^{\mathcal{D}}(s) \right) \right) \right]$
4. **end for**

---

Peng et al, 2019
I thought we were going
to talk about
transformers?
Transformers
Given sequence of English words, predict sequence of French
Transformer Architecture
"The animal didn't cross the street because it was too tired"
Attention as a soft-memory look up
Input

Embedding

Queries

Keys

Values

Score

Divide by $8 \left( \sqrt{d_k} \right)$

Softmax

Softmax $X$

Value

Sum

---

Thinking

- $x_1$:
  - $q_1$:
  - $k_1$:
  - $v_1$:

- $z_1$:

Machines

- $x_2$:
  - $q_2$:
  - $k_2$:
  - $v_2$:

- $z_2$:

- $q_1 \cdot k_1 = 112$
- $q_1 \cdot k_2 = 96$
- $14 / 8 = 1.75$
- $12 / 8 = 1.5$
- $0.88 / 8 = 0.11$
- $0.12 / 8 = 0.015$
Back to Decision Making
Decision Transformer: Reinforcement Learning via Sequence Modeling

Lili Chen*,1, Kevin Lu*,1, Aravind Rajeswaran2, Kimin Lee1, Aditya Grover2, Michael Laskin1, Pieter Abbeel1, Aravind Srinivas†,1, Igor Mordatch†,3

*equal contribution †equal advising

1UC Berkeley  2Facebook AI Research  3Google Brain

{lilichen, kzl}@berkeley.edu
\[ \hat{R} = \sum_{t=0}^{T-1} r_t \]

\[ \hat{R}_0 = \sum_{t=0}^{T-1} r_t \]

\[ \hat{R}_1 = \sum_{t=1}^{T-1} r_t \]
Introducing Decision Transformers on Hugging Face 😄
Test Time

Start at initial state $s_0$

Specify the desired target return $R_0$

$$a_0 = \text{Transformer}(R_0, s_0)$$

Execute action, observe reward and next state $(r_0, s_1)$

Decrement the target return $R_1 = R_0 - r_0$

$$a_1 = \text{Transformer}(R_0, s_0, a_0, R_1, s_1)$$
Seems to work!
Seems to work!

<table>
<thead>
<tr>
<th>Game</th>
<th>DT (Ours)</th>
<th>CQL</th>
<th>QR-DQN</th>
<th>REM</th>
<th>BC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakout</td>
<td>267.5 ± 97.5</td>
<td>211.1</td>
<td>17.1</td>
<td>8.9</td>
<td>138.9 ± 61.7</td>
</tr>
<tr>
<td>Qbert</td>
<td>15.4 ± 11.4</td>
<td>104.2</td>
<td>0.0</td>
<td>0.0</td>
<td>17.3 ± 14.7</td>
</tr>
<tr>
<td>Pong</td>
<td>106.1 ± 8.1</td>
<td>111.9</td>
<td>18.0</td>
<td>0.5</td>
<td>85.2 ± 20.0</td>
</tr>
<tr>
<td>Seaquest</td>
<td>2.5 ± 0.4</td>
<td>1.7</td>
<td>0.4</td>
<td>0.7</td>
<td>2.1 ± 0.3</td>
</tr>
</tbody>
</table>

Atari
Seems to work!

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Environment</th>
<th>DT (Ours)</th>
<th>CQL</th>
<th>BEAR</th>
<th>BRAC-v</th>
<th>AWR</th>
<th>BC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Medium-Expert</td>
<td>HalfCheetah</td>
<td>86.8 ± 1.3</td>
<td>62.4</td>
<td>53.4</td>
<td>41.9</td>
<td>52.7</td>
<td>59.9</td>
</tr>
<tr>
<td>Medium-Expert</td>
<td>Hopper</td>
<td>107.6 ± 1.8</td>
<td>111.0</td>
<td>96.3</td>
<td>0.8</td>
<td>27.1</td>
<td>79.6</td>
</tr>
<tr>
<td>Medium-Expert</td>
<td>Walker</td>
<td>108.1 ± 0.2</td>
<td>98.7</td>
<td>40.1</td>
<td>81.6</td>
<td>53.8</td>
<td>36.6</td>
</tr>
<tr>
<td>Medium-Expert</td>
<td>Reacher</td>
<td>89.1 ± 1.3</td>
<td>30.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>73.3</td>
</tr>
<tr>
<td>Medium</td>
<td>HalfCheetah</td>
<td>42.6 ± 0.1</td>
<td>44.4</td>
<td>41.7</td>
<td>46.3</td>
<td>37.4</td>
<td>43.1</td>
</tr>
<tr>
<td>Medium</td>
<td>Hopper</td>
<td>67.6 ± 1.0</td>
<td>58.0</td>
<td>52.1</td>
<td>31.1</td>
<td>35.9</td>
<td>63.9</td>
</tr>
<tr>
<td>Medium</td>
<td>Walker</td>
<td>74.0 ± 1.4</td>
<td>79.2</td>
<td>59.1</td>
<td>81.1</td>
<td>17.4</td>
<td>77.3</td>
</tr>
<tr>
<td>Medium</td>
<td>Reacher</td>
<td>51.2 ± 3.4</td>
<td>26.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>48.9</td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>HalfCheetah</td>
<td>36.6 ± 0.8</td>
<td>46.2</td>
<td>38.6</td>
<td>47.7</td>
<td>40.3</td>
<td>4.3</td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>Hopper</td>
<td>82.7 ± 7.0</td>
<td>48.6</td>
<td>33.7</td>
<td>0.6</td>
<td>28.4</td>
<td>27.6</td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>Walker</td>
<td>66.6 ± 3.0</td>
<td>26.7</td>
<td>19.2</td>
<td>0.9</td>
<td>15.5</td>
<td>36.9</td>
</tr>
<tr>
<td>Medium-Replay</td>
<td>Reacher</td>
<td>18.0 ± 2.4</td>
<td>19.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5.4</td>
</tr>
<tr>
<td>Average (Without Reacher)</td>
<td>74.7</td>
<td>63.9</td>
<td>48.2</td>
<td>36.9</td>
<td>34.3</td>
<td>46.4</td>
<td></td>
</tr>
<tr>
<td>Average (All Settings)</td>
<td>69.2</td>
<td>54.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>47.7</td>
<td></td>
</tr>
</tbody>
</table>

D4RL
Why does context length matter?

## K=50

<table>
<thead>
<tr>
<th>Game</th>
<th>DT (Ours)</th>
<th>DT with no context ($K = 1$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakout</td>
<td>267.5 ± 97.5</td>
<td>73.9 ± 10</td>
</tr>
<tr>
<td>Qbert</td>
<td>15.1 ± 11.4</td>
<td>13.6 ± 11.3</td>
</tr>
<tr>
<td>Pong</td>
<td>106.1 ± 8.1</td>
<td>2.5 ± 0.2</td>
</tr>
<tr>
<td>Seaquest</td>
<td>2.5 ± 0.4</td>
<td>0.6 ± 0.1</td>
</tr>
</tbody>
</table>
Concurrent paper

Offline Reinforcement Learning as One Big Sequence Modeling Problem

Michael Janner  Qiyang Li  Sergey Levine
University of California at Berkeley
{janner, qcli}@berkeley.edu  svlevine@eecs.berkeley.edu

Think of it as model-based RL / IL
Trajectory Transformer
Performs comparably to DT

The bar chart shows the average normalized return for different methods across various scenarios. The methods compared include:

- BC (Behavior Cloning)
- MBOP (Trajectory Optimization)
- BRAC (Temporal Difference)
- CQL (Sequence Modeling)
- Decision Transformer
- Trajectory Transformer

Each method is represented by a different color in the chart, allowing for easy comparison of performance.
Are we done?
Activity!
Consider the following MDP

\[
\begin{align*}
S_0 & \xrightarrow{a_0} S_1 \\
S_0 & \xrightarrow{a_1} S_2 \\
S_0 & \xrightarrow{a_2} R_1 \\
\end{align*}
\]

- \(S_0\) has a 50% chance of transitioning to a state with \(r = -5\) and another 50% chance of transitioning to a state with \(r = -15\)
- \(S_1\) has a 50% chance of transitioning to a state with \(r = 1\) and another 50% chance of transitioning to a state with \(r = -6\)
- \(S_2\) has a transition labeled \(r = 1\)
- \(R_1\) has a transition labeled \(r = 1\)
Consider the following MDP

What is the optimal action? What will Decision Transformer play?
Think-Pair-Share!

Think (30 sec): What is the optimal action? What would decision transformers play?

Pair: Find a partner

Share (45 sec): Partners exchange ideas
methods that condition on outcomes such as return can make incorrect decisions in stochastic environments regardless of scale or the amount of data they are trained on.
You Can’t Count on Luck: Why Decision Transformers and RvS Fail in Stochastic Environments

methods that condition on outcomes such as return can make incorrect decisions in stochastic environments regardless of scale or the amount of data they are trained on
But does it work in deterministic environments?
Activity!
Consider the following deterministic MDP
Data collection 1

$s_0 \xrightarrow{a_0} r = 0 \xrightarrow{a_0} r = -1$

100% $r = 0$

50% $r = 0$

50% $r = -1$
Data collection

100%

\[ a_1 \quad a_0 \quad r = 0 \]

\[ s_0 \quad s_2 \quad a_0 \quad r = -1 \]

\[ s_1 \quad \text{Grim Reaper} \quad a_1 \quad r = 0 \]
Let’s say we start from $s_0$

What will DT learn?

What will Q learning learn?
Let’s say we start from $s_0$

What will DT learn?

What if the context length $= 1$?

- $r = 0$
- $r = -1$
- $r = 0$
When does return-conditioned supervised learning work for offline reinforcement learning?

David Brandfonbrener  
New York University  
david.brandfonbrener@nyu.edu

Alberto Bietti  
New York University

Jacob Buckman  
MILA

Romain Laroche  
Microsoft Research

Joan Bruna  
New York University
Sufficient conditions for DT to work

Let’s data gathering policy be $\beta$, and $R^*(s)$ be the optimal return

Assume

1. Return coverage: $P_\beta(R = R^*(s_0)|s_0) \geq \alpha$ for all initial states $s_0$

   You will see all returns some fraction of the time from all initial states

2. Near determinism:

   $P(r \neq r(s,a) \text{ or } s' \neq T(s,a)|s,a) \leq \epsilon$ for all $(s,a)$

Then

$$J(\pi^*) - J(\pi_{DT}) \leq \epsilon \left( \frac{1}{\alpha} + 2 \right) H^2$$
Research Questions

Can we condition on better alternatives to return?

Train a value estimator (critic)