Numerical Goal-based Transformers for Practical Conditions

Seonghyun Kim, Samyeul Noh, Ingook Jang[*]
Electronics and Telecommunications Research Institute
218 Gajeong-ro, Daejeon, Korea
`{kim-sh, samuel, ingook}@etri.re.kr`

Abstract

Goal-conditioned reinforcement learning (GCRL) studies aim to apply trained agents in realistic environments. In particular, offline reinforcement learning is being studied as a way to reduce the cost of online interactions in GCRL. One such method is Decision Transformer (DT), which utilizes a numerical goal called "return-to-go" for superior performance. Since DT assumes an idealized environment, such as perfect knowledge of rewards, it is necessary to study an improved approach for real-world applications. In this work, we present various attempts and results for numerical goal-based transformers to operate under practical conditions.

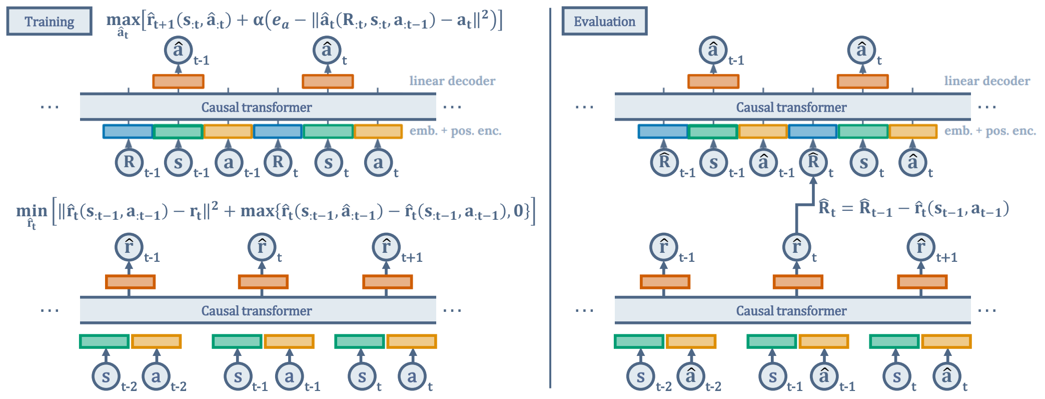
1 Motivation

In the field of reinforcement learning, various works are researched to apply agents to the real world. Goal-conditioned reinforcement learning (GCRL) is one of the research areas that focuses on this direction. GCRL studies are conducted in realistic environments that can serve a variety of goals, and various benchmarks and algorithms have been presented in this regard [1]-[3]. Previous works on GCRL have considered online interaction with the environment. However, the need for such online interaction with the environment can be costly and risky for real-world applications. To avoid the cost of interacting with the environment, offline reinforcement learning approaches that utilize previously collected offline datasets to train agents are gaining attention [4]-[7].

As one attempt to train agents using offline datasets, Decision Transformer (DT) that leverages sequence modeling problems is proposed [8]. In [8], it is shown that superior performance can be achieved by utilizing a numerical goal called return-to-go as a condition for a causal transformer. However, DT assumes an idealized environment for numerical goals, where the agent knows exactly the actual reward it receives from the environment at each step. This assumption is inconsistent with the fact that humans do not check rewards for every action in their lives. Although it is possible to utilize the diverse information in offline datasets to learn an agent's policy model in the training phase, it is necessary to consider practical conditions for real-world applications where learned models can operate with minimal information such as state or raw data.

In this work, we present various attempts and experimental results for numerical goal-based transformers to operate under practical conditions. In Section 2, the structure and objective functions of the proposed algorithm are described. In Section 3, performance comparisons between our proposed method and existing techniques are presented for various datasets and environments.

[*]corresponding author



Training
$\max_{\hat{a}_t}[\hat{r}_{t+1}(s_{:t}, \hat{a}_{:t}) + \alpha(e_a - \|\hat{a}_t(R_{:t}, s_{:t}, a_{:t-1}) - a_t\|^2)]$
linear decoder
Causal transformer
emb. + pos. enc.
$\min_{\hat{r}_t}[\|\hat{r}_t(s_{:t-1}, a_{:t-1}) - r_t\|^2 + \max\{\hat{r}_t(s_{:t-1}, \hat{a}_{:t-1}) - \hat{r}_t(s_{:t-1}, a_{:t-1}), 0\}]$
Evaluation
$\hat{R}_t = \hat{R}_{t-1} - \hat{r}_t(s_{t-1}, a_{t-1})$

Figure 1: **Conservative Decision Transformer (CDT):** CDT consists of two types of causal transformers. In the training phase, each transformer shares the trajectories in batches and updates network parameters according to its own objective function. In the evaluation phase, the transformers share and connect network outputs such as approximated action and reward, $\widehat{a}$ and $\widehat{r}$, respectively.

2 Methodology

Figure 1 represents the architecture of our proposed algorithm, Conservative Decision Transformer (CDT). In the training phase, all data from the offline dataset, including returns and rewards, are utilized to update the networks. In the evaluation phase, considering practical conditions, only states are used to interact with the environment.

2.1 Preliminaries

A Markov decision process is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ is a transition probability, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function. The numerical goal, return-to-go at time t, is given as $R_t = \sum_{k=t}^{T-1} r_{k+1}(s_k, a_k)$, where T is an episode horizon, and $r_{k+1}(s_k, a_k)$ is a reward for the state-action pair (s_k, a_k). Based on these notations, a trajectory is defined as $\tau = (R_0, s_0, a_0, r_1, R_1, ..., R_{T-1}, s_{T-1}, a_{T-1}, r_T)$.

2.2 Objective functions for Conservative Decision Transformer

The objective function for the reward estimation is defined as

$$J_\theta = \min_\theta \Big\{ \|\widehat{r}_t(s_{:t-1}, a_{:t-1}) - r_t\|^2 + \max\big(\widehat{r}_t(s_{:t-1}, \widehat{a}_{:t-1}) - \widehat{r}_t(s_{:t-1}, a_{:t-1}), 0\big) \Big\}, \tag{1}$$

where $\widehat{r}_t$ is an output of a parameterized network with parameter θ, $x_{:k}$ is a history data as $(x_0, ..., x_k)$, $\widehat{a}_t$ is a generated action not in an offline dataset and (s_t, a_t, r_t) are samples in an offline dataset.

Since the objective function in Eq. (1) is a minimization problem, the optimal value of the second term in Eq. (1) is 0. It means that $\widehat{r}_t(s_{:t-1}, \widehat{a}_{:t-1})$ should be lower than $\widehat{r}_t(s_{:t-1}, a_{:t-1})$, i.e., $\widehat{r}_t(s_{:t-1}, \widehat{a}_{:t-1}) \leq \widehat{r}_t(s_{:t-1}, a_{:t-1})$. This means that the estimated reward for unseen actions is conservatively lower than the reward for actions in the offline dataset.

The objective function for the action generation is defined as

$$J_\phi = \max_\phi \Big\{ \widehat{r}_{t+1}\big(s_{:t}, \widehat{a}_{:t}(R_{:t}, s_{:t}, a_{:t-1})\big) + \alpha\big(e_a - \|\widehat{a}_t(R_{:t}, s_{:t}, a_{:t-1}) - a_t\|^2\big) \Big\}, \tag{2}$$

where $\widehat{a}_t$ is an output of a parameterized network with parameter ϕ, α is a positive weight variable, and e_a is an action error threshold.

The objective function in Eq. (2) means that a generated action should be found by considering maximizing the estimated reward and minimizing the action error. When α is large, the objective function tends to minimize the action error, and when α is close to 0, it tends to maximize the estimated reward. The weight α is automatically adjusted by solving a Lagrangian dual problem.

Table 1: Normalized returns for all algorithms in Mujoco locomotion tasks. The highest values for all algorithms are highlighted in cyan and the highest values for the proposed algorithms are highlighted in bold.

Dataset	Environment	CDT e_a =0.4	CDT e_a =0.38	DT-CR	DT-R	DT	CQL
Medium Expert	HalfCheetah	**91.8**	91.3	86.9	85.6	86.8	62.4
	Hopper	**107.7**	107.5	105.2	91.0	107.6	111.0
	Walker2d	107.2	**107.9**	107.0	106.7	108.1	98.7
Medium	HalfCheetah	40.1	**40.8**	**40.8**	40.0	42.6	44.4
	Hopper	62.4	**66.1**	63.3	59.5	67.6	58.0
	Walker2d	62.7	**72.5**	71.2	70.4	74.0	79.2
Medium Replay	HalfCheetah	31.6	33.1	**35.7**	32.9	36.6	46.2
	Hopper	62.9	73.7	**77.2**	69.5	82.7	48.6
	Walker2d	55.0	58.7	**59.7**	59.0	66.6	26.7

For the objective function in Eq. (2), a constrained optimization problem can be derived as

$$\begin{aligned} &\max_{\phi} \ \widehat{r}_{t+1}(s_{:t}, \widehat{a}_{:t}) \\ &\text{s.t.} \ \|\widehat{a}_t(R_{:t}, s_{:t}, a_{:t-1}) - a_t\|^2 \le e_a. \end{aligned}$$

The constrained optimization problem means that the parameter ϕ is found to maximize the estimated reward $\widehat{r}$ within the action error threshold e_a. A dual problem for the constrained optimization problem is defined as

$$J_\alpha = \min_{\alpha} \ \widehat{r}_{t+1}\big(s_{:t}, \widehat{a}_{:t}(R_{:t}, s_{:t}, a_{:t-1})\big) + \alpha\big(e_a - \|\widehat{a}_t(R_{:t}, s_{:t}, a_{:t-1}) - a_t\|^2\big), \tag{3}$$

where α is the Lagrange multiplier. Under the consideration of parameterized outputs $\widehat{r}$ and $\widehat{a}$ with fixed parameters θ and ϕ, the estimated reward $\widehat{r}_{t+1}$ in Eq. (3) does not affect to the Lagrange multiplier. Then the objective function in Eq. (3) can be simplified as

$$J_\alpha = \min_{\alpha} \alpha\big(e_a - \|\widehat{a}_t(R_{:t}, s_{:t}, a_{:t-1}) - a_t\|^2\big). \tag{4}$$

Since the objective function in Eq. (3) is a minimization problem, α is going to 0 for the condition $\|\widehat{a}_t(R_{:t}, s_{:t}, a_{:t-1}) - a_t\|^2 \le e_a$, which means that the generated action is within the threshold range of the action in the offline dataset. For the opposite condition, α is going to be larger.

3 Experimental Results

All experimental results for different datasets and environments are summarized in Table 1, where CQL and DT numbers are reported from the original papers [6] and [8], respectively. As shown in Table 1, for most datasets and environments, the proposed algorithms have similar performance with slightly lower values than those of DT due to estimation errors. For the Medium Expert dataset, it is observed that CDT has better or comparable performance to DT. However, for the Medium and Medium Replay datasets, it is observed that the performance gap between CDT and DT gradually increases. These results show that, from the perspective of the quality of the dataset, the performance of the proposed algorithm decreases as the proportion of highly rewarded trajectories in the overall dataset decreases and the consistency of actions decreases.

Comparing DT-R and DT-CR over the entire datasets and environments, it is shown that the reward estimation to minimize the error in a conservative manner outperforms the reward estimation to minimize the error only. For the Medium Expert and Medium datasets, comparing CDT and DT-CR, it is observed that CDT has higher performance because it generates actions by considering the maximization of the conservatively estimated reward. However, for the Medium Replay dataset, it is observed that DT-CR outperforms CDT. This means that if the dataset has few trajectories with high rewards and low consistency of actions in trajectories, it is better to only minimize the action error rather than maximize the conservatively estimated reward. In other words, a conservatively estimated reward on this dataset is susceptible to OOD problems.

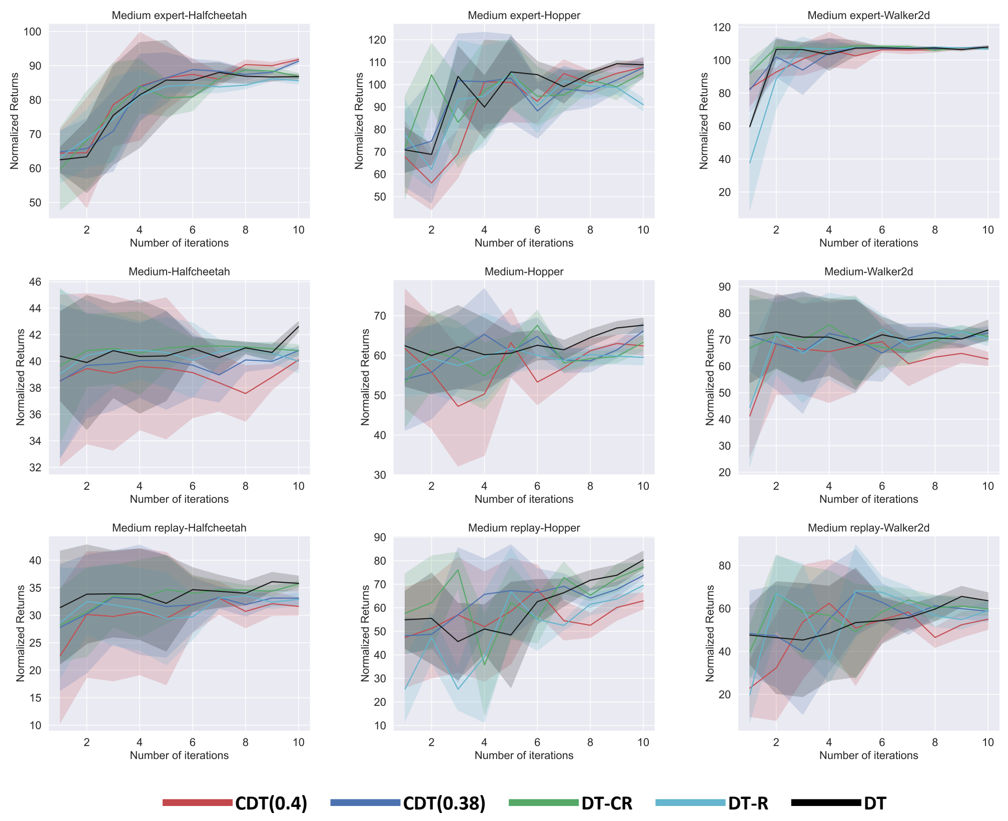


Figure 2: Training curves of all algorithms in Mujoco locomotion tasks.

When actions are highly consistent, such as in the Medium Expert dataset, CDT shows that there is no significant difference in performance according to the action error threshold. However, if the actions are somewhat less consistent, such as in the Medium and Medium Replay datasets, it is observed that CDT has better performance when setting the action error threshold conservatively. This is consistent with the results that DT-CR, which does not consider reward maximization but only action errors, outperforms CDT on the Medium Replay dataset.

Figure 2 represents the training curves of all algorithms. Each iteration consists of 10000 steps, and Table 1 is organized based on the performance of the 10th iteration [1]. Overall results show that most of the performance of the proposed algorithms converges around DT. As shown in Figure 2, from the Medium Expert to the Medium Replay of datasets, CDT with $e_a = 0.4$ has higher variances than those of CDT with $e_a = 0.38$. It means that this high variance of CDT with $e_a = 0.4$ due to relaxed error threshold has positive effects on good quality datasets, such as the Medium Expert dataset, but on the other hand, it is difficult to achieve positive effects on slightly lower quality datasets, such as the Medium and Medium Replay datasets.

4 Conclusion

In this work, we propose CDT for numerical goal-based transformers to operate in practical environments. Experimental results show that the CDT can achieve stable performance with only state information and no actual reward information. In future work, we would like to study a generalized GCRL for 3D locomotion and robot manipulation tasks by considering various types of goals such as images, text, symbols, etc.

[1]Since the graphs of DT are run by us, the values of the 10th iteration are slightly different from those of Table 1, which are directly reported from [8].

Acknowledgments and Disclosure of Funding

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government. [23ZR1100, A Study of Hyper-Connected Thinking Internet Technology by autonomous connecting, controlling, and evolving ways].

References

[1] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," Advances in neural information processing systems, vol. 30, 2017.

[2] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in International conference on machine learning. PMLR, 2018, pp. 1515–1528.

[3] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp, "Goal-conditioned imitation learning," Advances in neural information processing systems, vol. 32, 2019.

[4] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," arXiv preprint arXiv:2005.01643, 2020.

[5] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," arXiv preprint arXiv:2004.07219, 2020.

[6] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," Advances in Neural Information Processing Systems, vol. 33, pp. 1179–1191, 2020.

[7] D. Ghosh, A. Gupta, A. Reddy, J. Fu, C. Devin, B. Eysenbach, and S. Levine, "Learning to reach goals via iterated supervised learning," arXiv preprint arXiv:1912.06088, 2019.

[8] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," Advances in neural information processing systems, vol. 34, pp. 15 084–15 097, 2021.

A Conservative Decision Transformer Algorithm

Based on the objective functions in Eqs. (1), (2) and (4), the parameters θ, ϕ and α are updated by a recursive optimization as shown in Algorithm 1.

Algorithm 1 Conservative Decision Transformer

```
 1: Initialize parameters $\theta$, $\phi$ and $\alpha$.
 2: Load offline dataset $\mathcal{D} = \{(R_t, s_t, a_t, r_{t+1}) | (R_t, s_t, a_t, r_{t+1})$ over all t in all episodes.}
 3: for each iteration do
 4:    Sample $K$ trajectories with batch size $B$:
 5:       $\mathcal{D}_{K,B} \sim \mathcal{D}(R_{t:t+K-1}, s_{t:t+K-1}, a_{t:t+K-1}, r_{t+1:t+K})$
 6:    for each gradient step do
 7:       Update the reward estimation:
 8:          $\theta \leftarrow \theta - \lambda \nabla_\theta J_\theta$ using Eq. (1)
 9:       Update the action generation:
10:          $\phi \leftarrow \phi - \lambda \nabla_\phi J_\phi$ using Eq. (2)
11:       Update the weight:
12:          $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha J_\alpha$ using Eq. (4)
13:    end for
14: end for
```

B Experimental Details

In this section, we describe experimental details. The algorithm implementation is based on DT [8].

B.1 Network architectures

Our two types of network architectures are the same as DT, where the number of layers is 3, the number of attention heads is 1, the embedding dimension is 128, and the nonlinearity function is ReLU for each network.

B.2 Hyperparameters

Hypermaraters are summarized in Table 2.

Table 2: Hyperparameters of CDT

Hyperparameters	Value
batch size	64
context length K	20
learning rate for $\widehat{r}$ and $\widehat{a}$	10^{-4}
learning rate for α	$2 \cdot 10^{-5}$
weight decay	10^{-4}
gradient norm clip	0.25
dropout	0.1
initial return-to-go	$6 \cdot 10^3$ for HalfCheetah
	$3.6 \cdot 10^3$ for Hopper
	$5 \cdot 10^3$ for Walker
action error threshold	0.38, 0.4