DOMAIN TRANSFER WITH LARGE DYNAMICS SHIFT IN OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

The domain transfer problem with large dynamics shift commonly exists when using offline reinforcement learning (RL) in real-world applications, where the source dataset collected from one domain needs to be reused to accelerate training the target domain agent with offline RL. The large dynamics shift issue arises when there are unpredictable changes in the target domain's environment. Existing works typically assume that each state-action pair in the target domain should be at least covered in the source domain, which is often unrealistic and limited to small dynamics shift transfers. To tackle the large dynamics shift problem, we propose to use the source domain data not only for offline policy training but also for safe and efficient data collection in the target domain, thus relaxing the above requirement. Specifically, the source data will play two roles, one is to serve as augmentation data by compensating for the difference in dynamics with modified reward. Another is to form prior knowledge for the behaviour policy to collect a small amount of new data in the target domain safely and efficiently. The target domain policy is trained using offline RL with the source data and modest amounts of newly collected target data. We justify our method in gridworld and autonomous driving environments. Results show that our method requires fewer target domain data and collecting the data in a safer manner compared with prior methods.

1 INTRODUCTION

Reinforcement learning (RL) is a paradigm of learning from interaction, so as to maximize the accumulated reward signal (Sutton & Barto, 2018). Combining with deep neural networks (Goodfellow et al., 2016), DRL shows powerful capabilities on various decision-making problems, including video games (Mnih et al., 2015; Hessel et al., 2018), robotic control (Lillicrap et al., 2015; Haarnoja et al., 2018) and two-player zero-sum board games (Silver et al., 2016; 2017). Though this trial-and-error paradigm achieves huge success in domains which are cheaper to collect data, there are many domains where collecting data is costly and laborious. For example, in autonomous driving, trials should be cautious because the errors are too costly. For healthcare scenarios, the data for a particular patient is often limited. These real problems push us to make RL algorithms safe and efficient. Relief is that there are many different but similar tasks we call them source domains. We can help the learning on a target domain by transferring the knowledge contained in different but related source domains (Zhuang et al., 2020).

In this paper, we design a transfer method for large dynamics shift in offline RL. The motivation is that if a similar transition can be obtained from source data, we can make use of it and don't need to collect that part of the data in the target domain. And when we collect data in the target domain, we should visit transitions with a large dynamic shift in the target domain compared with the source domain and also avoid unsafe actions. Our method makes most of the source data in two aspects. One is to use source data as data augmentation for the target domain with a modified reward that compensates for the dynamics shift. Another is to compose prior knowledge for safe and efficient data collecting in the target domain. Figure 1 gives an illustration of our method. We have a source domain and a target domain with different dynamics. We construct the target domain using source data and target data. Source data with modified reward provide similar transitions in both domains, and target data mainly contribute to the part that has a big difference.

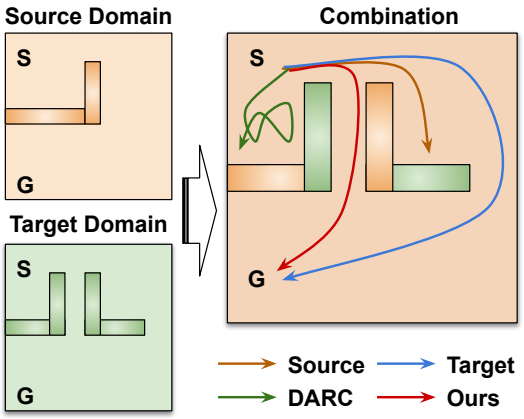


Figure 1: We make full use of source data and a small amount of target data to recover the dynamics for target domain. Our method learns the optimal policy with only limited target domain data collection.

In dynamics shift setting, the only difference between source domain and target domain is the dynamics, $p_{\text{source}}(s_{t+1}|s_t, a_t)$ and $p_{\text{target}}(s_{t+1}|s_t, a_t)$. Previous works assume that each state-action pair in the target domain should be at least covered in the source domain, which is often unrealistic and limited to small dynamics shift transfers. Without this assumption, the large dynamics shift will misguide the learning to a suboptimal policy. As shwon in Figure 1, the DARC (Eysenbach et al., 2021) algorithm can find a path that exists in source domain, but failed to find the optimal path that only exists in target domain. Our method relaxes this assumption and only demands that some part of the source and target domains are covered. And the uncovered part is made up by a behaviour policy with limited interation with target domain. Specifically, assume that we have access to a large offline dataset collected from a source domain and a small amount of initial target data. We first train a pair of classifiers to distinguish source and target transitions using supervised learning. Then, based on the classifiers, we use the offline source data for two purposes: (1) Since the dynamics are different between source domain and target domain. The source data cannot directly be used to learning a policy for target domain. We use the classifiers to modify the reward for the source data by adding Δr to compensate the dynamics difference. (2) For the transitions that never appear in the source domain, we need to collect that part of data in the target domain. We predict the source data with classifiers, if a transition may also appear in the target domain with high probability, we store it into a graph structure to compose prior knowledge. A behaviour policy is designed based on the prior knowledge to collect a small amount of new data in the target domain. The prior knowledge will guide the behaviour policy to visit unfamiliar transition in a safe manner. Combining the two usages, the source data with modified reward and the small amount of newly collected target data will be used together to learn a policy for target domain with offline methods.

Our contribution is four-fold: (1) We propose to combine source data with a small amount of target data for large dynamics shift transfer in offline RL. (2) We augment target data with reward modification for source data with a pair of learned domain classifiers. This data augmentation technique improve sample efficiency to a large margin. (3) We design a behaviour policy with prior knowledge based on source data for target domain. The prior knowledge will guide the behaviour policy to visit unfamiliar transitions in a safe manner, which achieves safe and efficient data collection in target domain. (4) We justify our method on gridworld and autonomous driving environments. Results show that our method collects data in a safer manner and achieving higher sample efficiency comparing with prior methods.

2 Related Work.

Offline Reinforcement Learning. Offline reinforcement learning (also known as batch RL) describes the setting that online interaction with environments is not allowed. The goal is to learn a highly rewarding policy (target policy) only using previously collected data (by behavior policy) (Lange et al., 2012; Levine et al., 2020). Consider that online interaction is impractical for many real-world problems due to high expense and safety risk (e.g. robotics, autonomous driving and healthcare), offline RL, the pure data-driven learning of policies is a more practical setting.

Without further interaction with environments, one unique challenge for offline RL is the distributional shift between the policy that collected the data (behavior policy) and the learned policy (target policy). It means a policy might be trained under one distribution but evaluated on a different distribution (Fujimoto et al., 2019; Levine et al., 2020). Offline RL methods address this problem by constraining the target policy to be closer to the behavioral policy. Common methods include constraining the difference between the target policy and behavior policy, such as BCQ (Fujimoto

et al., 2019), BEAR (Kumar et al., 2019) and AWAC (Nair et al., 2020). Other methods achieve this by doing conservative value update based on penalty term or ensembled Q function such as CQL (Kumar et al., 2020) and REM (Agarwal et al., 2020).

Dynamics Shift Transfer in offline RL. One cause for distributional shift is dynamics shift. Dynamics shift describe the situation that the dynamics of source domain/data is different from that of target domain/data. The source domain can be a simulator or some part of the environment, the target domain can be the real world. Transfer learning has been a popular and promising topic in machine learning (Pan & Yang, 2009) and deep learning (Tan et al., 2018). It aims at fast adaptation and performance improving on target domains by transferring the knowledge contained in different but related source domains (Zhuang et al., 2020). The same demand exists in offline RL. RL tends to be task specific and suffers from low sample efficiency. A more realistic problem is that the interaction with environments is impractical for many real-world problems due to high expense and safety risk such as for robotics, autonomous driving and healthcare (Li, 2017). These challenges highlight the demand of combining transfer learning with offline RL (Taylor & Stone, 2009; Zhu et al., 2020).

To handle this setting, DARC (Eysenbach et al., 2021) learns auxiliary classifiers to distinguish transitions from source domain and target domain. With a reward penalty based on the classifiers, agent is encouraged to learn a policy that performs well in both domains, especially in target domain. DARS (Liu et al., 2021) proposes to learn transferrable skills with an unsupervised domain adaptation method. It introduces a KL regularized objective to identify and acquire adaptive skills across dynamics. DARA (Liu et al., 2022) extends the learning of auxiliary classifiers to both model-free and model-based offline settings. Sasso et al. (2022) propose fractional transfer learning and universal feature spaces from a universal autoencoder (Chen et al., 2021) based on Dreamer (Hafner et al., 2020). This method autonomously extracts relevant knowledge and resulting in substantial performance improvements compared to learning from scratch. Different from these works that focus on representation or skills transfer, our method focuses on the full use of source data and a small amount of target data with transfer learning to tackle big dynamics shift.

3 BACKGROUND

Markov Decision Process (MDP). The interaction of an agent with the environment can be modeled as a MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, P, \rho_0, \gamma)$. $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the environment transition dynamics, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}$ is the initial state distribution and $\gamma \in (0, 1)$ is the discount factor. The goal of the agent is to learn the optimal policy that maximizes the expected cumulative rewards $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$. We have Q function $Q(s, a) := \mathbb{E}_\pi[\sum_{t=h}^{\infty} \gamma^{t-h} r_t | s_h = s, a_h = a]$, value function $V(s) := \mathbb{E}_{a \sim \pi(a|s)}[Q(s, a)]$, to measure the cumulative rewards under policy π.

Dynamics shift in RL. We consider two different MDPs. $\mathcal{M}_{\text{source}}$ represents the source domain, it can be a simulator, a similar but different real scenario, or a learned model. $\mathcal{M}_{\text{target}}$ represents the target domain, it can be the real world or a task that is too costly to collect enough data. $\mathcal{M}_{\text{target}}$ is the domain we want to learn an optimal policy for. We assume the only difference between the two domains is the dynamics P and other elements are the same. Though P_{source} is different from P_{target}, experience in the source domain is much cheaper to collect. Different from DARC (Eysenbach et al., 2021) that assumes any transition in target domain must appears in the source domain, i.e. $p_{\text{target}}(s_{t+1}|s_t, a_t) > 0 \Rightarrow p_{\text{source}}(s_{t+1}|s_t, a_t) > 0$, for all $s_t, s_{t+1} \in \mathcal{S}, a_t \in \mathcal{A}$. We relax the assumption that not all but some of the transitions in target domain should appear in the source domain, that is for all $s_t, s_{t+1} \in \mathcal{S}, a_t \in \mathcal{A}$, we have:

$$\{(s_t, a_t, s_{t+1}) | p_{\text{target}}(s_{t+1}|s_t, a_t) > 0, p_{\text{source}}(s_{t+1}|s_t, a_t) > 0, \} \neq \emptyset. \tag{1}$$

The intuition here is to guarantee the two domains to be at least similar in some parts, thus there exists related knowledge. Otherwise, there is no knowledge to transfer since the two domains are completely different. In offline RL setting, we have access to a static dataset $\mathcal{D} = \{(s, a, r, s')\}$ collected by one or mixture of k behavior policies $\{\beta_1, \beta_2, \cdots, \beta_k\}$. The goal is to learn the best possible policy using the offline dataset collected by the behaviour policies. Combining offline RL with transfer learning, suppose we have two offline datasets $\mathcal{D}_{\text{source}}$ and $\mathcal{D}_{\text{target}}$ from source domain $\mathcal{M}_{\text{source}}$ and target domain $\mathcal{M}_{\text{target}}$. We aim to learn a best possible policy for target domain $\mathcal{M}_{\text{target}}$ by using data $\mathcal{D}_{\text{source}}$ and $\mathcal{D}_{\text{target}}$ properly. In our setting, we can further collect a few amounts of

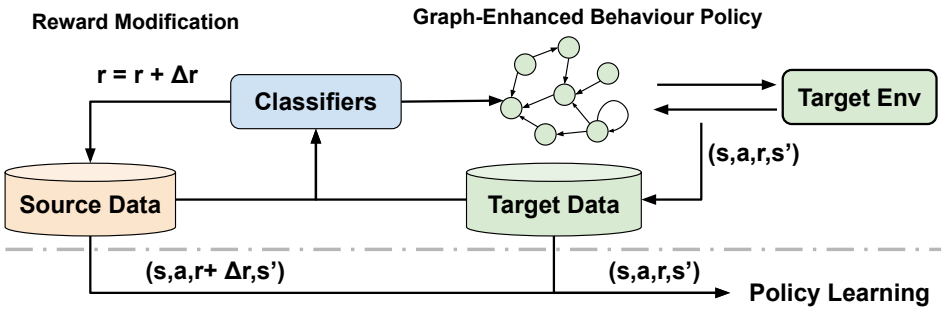


Figure 2: An illustration of our method. At each data collection iteration, we update classifiers with source and target data. We modify the reward for target data, and design a graph-enhanced behaviour policy to collect a small amount of new data in target environment. Both source and target data are used for policy learning.

data from target domain with behavior policy β_k at iteration k. This allows us to supply some data that is not accessible in source domain.

4 METHOD

Our intuition is to make full use of source data to help improve sample efficiency and relax the requirement for the target data. Our method first learns two classifiers to distinguish transitions from source domain and target domain. The classifiers is used to compensate the dynamic shift for transitions in source domain by modifying rewards. Then, also based on the classifiers, we build a graph structure to construct prior knowledge for target domain. The prior knowledge is used to guide the behaviour policy to newly collect a small amount of data in target domain. During policy learning, both transitions from source domain and target domain are used for learning. In Section 4.1, we introduce the way we learn the classifiers, and then use the classifiers to modify the reward for source offline data. In Section 4.2, we describe how we build a graph to construct prior knowledge and use it to guide the behaviour policy to collect data in target domain. In Section 4.3, we combine the two usages with offline RL algorithms and summarize the whole method.

4.1 DATA AUGMENTATION WITH REWARD MODIFICATION

Domain Classifiers. We learn a pair of classifiers to distinguish state-action pairs and state-action-next-state pairs respectively. Similar to DARC (Eysenbach et al., 2021), let $q_{\theta_{\text{SA}}}(\text{target}|s_t, a_t)$ and $q_{\theta_{\text{SAS}}}(\text{target}|s_t, a_t, s_{t+1})$ be the two classifiers parameterized by θ_{SA} and θ_{SAS}. We minimize the cross-entropy loss:

$$\begin{aligned}\mathcal{L}_{\text{SA}} &= -\mathbb{E}_{\mathcal{D}_{\text{target}}}[\log q_{\theta_{\text{SA}}}(\text{target}|s_t, a_t)] - \mathbb{E}_{\mathcal{D}_{\text{source}}}[\log q_{\theta_{\text{SA}}}(\text{source}|s_t, a_t)] \\ \mathcal{L}_{\text{SAS}} &= -\mathbb{E}_{\mathcal{D}_{\text{target}}}[\log q_{\theta_{\text{SAS}}}(\text{target}|s_t, a_t, s_{t+1})] - \mathbb{E}_{\mathcal{D}_{\text{source}}}[\log q_{\theta_{\text{SAS}}}(\text{source}|s_t, a_t, s_{t+1})]\end{aligned} \tag{2}$$

In our large dynamics shift setting, besides the initial target data, we are allowed to collect a small amount of new data in the target domain. During the process, the two classifiers will also be updated with newly collected data.

Reward Modification. Since the dynamics are different between source domain and target domain. The source data cannot directly be used to learning a policy for target domain. We use the classifiers to modify the reward for the source data by adding Δr to compensate the dynamics difference. Δr is defined as:

$$\Delta r(s_t, a_t, s_{t+1}) \triangleq \log p_{\text{target}}(s_{t+1}|s_t, a_t) - \log p_{\text{source}}(s_{t+1}|s_t, a_t). \tag{3}$$

Using Bayes' rule and the domain classifiers, we have:

$$\Delta r(s_t, a_t, s_{t+1}) = \log \frac{q_{\theta_{\text{SAS}}}(\text{target}|s_t, a_t, s_{t+1})}{q_{\theta_{\text{SAS}}}(\text{source}|s_t, a_t, s_{t+1})} - \log \frac{q_{\theta_{\text{SA}}}(\text{source}|s_t, a_t)}{q_{\theta_{\text{SA}}}(\text{target}|s_t, a_t)}. \tag{4}$$

Then the transition in the source data is modified as $(s_t, a_t, r + \Delta r, s_{t+1})$. This fomulation is similar to DARC (Eysenbach et al., 2021), we leave the derivation in Appendix A.1. From the definition of Δr in equation 3, this reward modification only applies when transitions appear in both source domain and target domain.

4.2 SOURCE DATA-INFORMED TARGET DOMAIN DATA COLLECTION

Algorithm 1 Behaviour Policy for Data Collection

Input: Graph $\mathcal{G}$, embedding function ϕ, source data $\mathcal{D}_{\text{source}}$, target data $\mathcal{D}_{\text{target}}$
Input: Learned classifier $q_{\theta_{\text{SAS}}}$, learned policy π_θ
Output: Target data $\mathcal{D}_{\text{target}}$

/ * Graph Building * /
for transition (s, a, r, s') in $\mathcal{D}_{\text{source}}$ **do**
  if $q_{\theta_{\text{SAS}}}(\text{target}|s_t, a_t, s_{t+1}) > \eta$ **then**
    Add (s, a, r, s') to $\mathcal{G}$, $N(s, a, s') \leftarrow N(s, a, s') + 1$
  end if
end for
Compute $\hat{r}$ via equation 7 and $Q(s, a)$ via equation 8 for all transitions

/ * Data Collection * /
Initialize the target environment $s_0 \leftarrow Env$
for environment step $t = 0$ **to** T **do**
  if $\phi(s_t) \in \mathcal{G}$ **then**
    $a_t = \arg\max Q_{a_t:\beta_{\text{source}}(a_t|s_t)>0}(s_t, \cdot)$
  else
    $a_t = \pi_\theta(s_t)$
  end if
  Execute a_t in Env and get r_t, s_{t+1}
  Add (s_t, a_t, r_t, s_{t+1}) to $\mathcal{G}$ and $\mathcal{D}_{\text{target}}$, update $N(s_t, a_t, s_{t+1})$, update $\hat{r}_t$ via equation 7
  Update $Q(s, a)$ via equation 8 with a sample batch
end for

For transitions that only exist in target domain, we design a behaviour policy to newly collect a small amount of data in target domain. To collect data safely and efficiently in target domain, we make use of source data to construct prior knowledge for the behaviour policy. We connect source data using a directed graph, the graph will give us an empirical environment dynamics and a conservative estimation. Then, a behaviour policy is designed on top of the graph.

Graph Building. We build a directed graph to save source transitions that may also appear in target domain. Along with transitions, we compute the corresponding statistics. We define the graph $\mathcal{G}$ as

$$\mathcal{G} = (V, E), V = \{\phi(s)|(s, \hat{Q}(s, \cdot))\}, E = \{\phi(s) \xrightarrow{a} \phi(s')|(a, \hat{r}, N(s, a, s'))\}. \tag{5}$$

Here, V denotes the set of vertices and E denotes the set of egdes. V contains states and action value estimation $Q(s, \cdot)$. ϕ is an embedding function to map the high-dimensional states to a low-dimensional representation for fast query (keys in the graph $\mathcal{G}$). We store vertices in a dictionary, and with ϕ as a hashing function to serve as keys, thus we can retrieve a state in $\mathcal{O}(1)$ time. For environments with continuous state and action spaces, we first employ the K-bins discretization technique (Kotsiantis & Kanellopoulos, 2006) to merge similar actions and states. $Q(s, \cdot)$ only contains values for actions that were taken by behavior policy in source domain β_{source}, i.e., $a : \beta_{\text{source}}(a|s) > 0$. E is identified by the tuple $(\phi(s), a, \phi(s'))$, each edge contains statistics of action a, reward $\hat{r}$ and visit count $N(s, a, s')$. These statistics are similar to the ones that used in Monte Carlo Tree Search (MCTS) of AlphaGo (Silver et al., 2016; 2017; 2018; Schrittwieser et al., 2020), where statistics are stored in a tree. With a discount factor smaller than 1, our graph has no limitation for the structure. For example, the graph can have cycles and dynamics can be stochastic. Then we estimates the empirical dynamics $\hat{P}$ at transition (s_t, a_t, s_{t+1}) with visit counts:

$$\hat{p}(s_{t+1}|s_t, a_t) = \frac{N(s_t, a_t, s_{t+1})}{\sum_{s_{t+1}} N(s_t, a_t, s_{t+1})}. \tag{6}$$

Before we add a source transition to the graph, we first use the classifier $q_{\theta_{\text{SAS}}}(\text{target}|s_t, a_t, s_{t+1})$ to discriminate if this transition may appears in the target domain. We set a probability threshold η, if $q_{\theta_{\text{SAS}}}(\text{target}|s_t, a_t, s_{t+1}) > \eta$, this transition will be added to the graph. Otherwise, the transition

will be discarded. When we add a transition to the graph, we first check if $\phi(s)$ exists in the graph. If not, we create a new vertex and edge and initialize $Q(s,a) = 0, N(s,a,s') = 1$. Otherwise, if $\phi(s)$ already exists, we merge the same states into one node and update the visit count $N(s,a,s') = N(s,a,s') + 1$. It means different trajectories that have the same state representation will intersect with each other. To collect data more efficient, we add a count-based intrinsic reward based on the state visits,

$$\hat{r} = r_{\text{env}} + \alpha r_{\text{intrinsic}}, \tag{7}$$

r_{env} is the reward from the environment, $r_{\text{intrinsic}} = \frac{\sqrt{\sum_s N(s)}}{1+N(s)}$ will encourage visiting less visited states, and α is a parameter. This formula is similar to UCB (Kocsis & Szepesvári, 2006) algorithm that commonly used in MCTS (Browne et al., 2012), which makes a trade-off between exploitation and exploration.

Behaviour Policy for Data Collection. We first solve the graph to form prior knowledge. With all these statistics, the graph structure forms a MDP with finite states and actions. Then tabular method can be used directly. We use value iteration to update Q:

$$Q(s,a) \leftarrow \sum_{s'} \hat{p}(s'|s,a)[\hat{r} + \gamma \max_{a': \beta_{\text{source}}(a'|s') > 0} Q(s',a')], \tag{8}$$

where $\hat{p}$ is the empirical dynamics based on the graph computed by equation 6, γ is discount factor, $Q(s',\cdot)$ and $\hat{r}$ can be queried from corresponding vertex and edge in $\mathcal{O}(1)$ time. Since there is a constraint $a' : \beta_{\text{source}}(a'|s') > 0$ when we update Q based on equation 8. This will exclude unvisited actions when computing maximum Q. This means the Q value will give us a conservative estimation that considers both safety and performance. This is a good prior knowledge to construct a behaviour policy for target domain.

Then the behaviour policy is a combination of the Q function estimated from the graph and the current learned policy. When we collect data in the target domain, we first inquiry if the representation of current state $\phi(s)$ is in the graph. If it is, we choose the action with maximum Q value, otherwise we use the current learned policy to choose actions. Remember that, Q only includes existed actions, this will avoid dangerous unvisited actions. And the intrinsic reward nested in r will help collect data more efficient. During the data collection, each interaction step will add a new transition to the graph. We need to update Q simultaneously. We use a sampling-based method to update Q instead of updating the whole graph at each step. Similar to experience replay (Lin, 1992), we sample a batch from the graph and update Q with equation 8. Besides, when an episode ends, we update Q along the episode in a reversed manner, which is similar to EBU (Lee et al., 2019) algorithm. With the sampling-based update and episode-based update, the two techniques allow sparse and delayed rewards to propagate quickly through all transitions of related trajectories, and the computation complexity is controlled in a low degree. We summarize the process in Algorithm 1.

4.3 Algorithm Summary

Then, we use the transitions both in the source domain $(s_t, a_t, r + \Delta r, s_{t+1})$ and target domain (s_t, a_t, r, s_{t+1}) to learn a policy that performs well in the target domain. For each data collection iteration, we first update the classifiers, then update the transitions in the graph and collect data in the target domain with prior knowledge computed based on the graph, and finally we modify the reward of source data and update the policy using both source and target data. We summarize the overall method in Algorithm 2.

Algorithm 2 Domain Transfer with Large Dynamics Shift

1: Initialize graph $\mathcal{G}$, embedding function ϕ, source data $\mathcal{D}_{\text{source}}$, target data $\mathcal{D}_{\text{target}}$
2: Initialize classifiers $q_{\theta_{\text{SAS}}}$ and $q_{\theta_{\text{SA}}}$, policy π_θ
3: **for** data collection iteration $k = 0$ **to** K **do**
4: Train $q_{\theta_{\text{SAS}}}$ and $q_{\theta_{\text{SA}}}$ with $\mathcal{D}_{\text{source}}$ and $\mathcal{D}_{\text{target}}$
5: Build graph $\mathcal{G}$ and collect data in target domain via Algorithm 1
6: Modify rewards for source data via equation 4
7: Update policy π_θ with any offline RL method using $\mathcal{D}_{\text{source}}$ and $\mathcal{D}_{\text{target}}$
8: **end for**

5 EXPERIMENTS

In this section, we aim to answer the following questions: (1) Does our method improve performance compared with previous works? (2) What are the effects of reward modification? (3) Does the graph-enhanced behaviour policy collect data safely and efficiently?

5.1 A TOY EXAMPLE

We first give a toy example to illustrate that our method can learn an optimal policy efficiently. Consider gridworld environments as shown in Figure 1. The source domain and target domain contain different obstacles. The goal is to start from the top left, bypass obstacles and finally reach the goal state at the bottom left. The reward of reaching the goal is +10. The max steps is set to 500, and each step have a penalty reward -0.1 so that an agent will get -50 if it does not reach the goal. In our experiments, the source offline data has 100k transitions and target data has 10k transitions. We additionally allow 10k interactions with target domain. We compare our methods with **Source**, **Target** and DARC algorihms. **Source** learns a policy only with source data. **Target** learns a policy only with target data. DARC uses both source and target data, and applies reward modification technique. Our method uses both reward modification and graph-enhanced data collection techniques.

We report two metrics in Table 1. Final score shows the final performance for each method. Collisions shows the number of times the agent collide into walls or obstacles, which indicates unsafe actions in target domain. We can find our method achieves the best performance. DARC reaches the goal finally but with a suboptimal path. **Source** fails to reach the goal and collides into the obstacle due to unwareness of the dynamics difference. **Target** fails to reach the goal due to limited interaction with the target domain. Our method combines both source data and target data and construct the dynamics for target domain. This method can take use of similar source data and also make the best use of liminted interactions with target domain. We show the combination dynamics and the policy trajectory for each method in Figure 1 for clear comparison. In addition, from the number of collisions in Table 1, we can easily find that with the graph-enhanced behaviour policy, our method collects data in a safer manner.

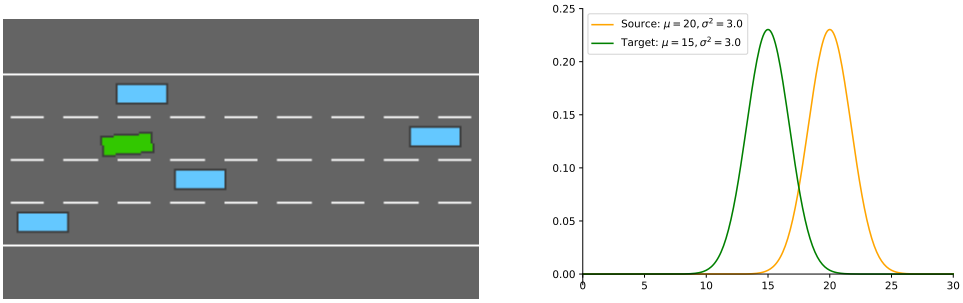
Table 1: Performance on gridworld environment.

	Source	Target	DARC	Ours
Final Score	-50	-50	7.3	8.3
Collisions	0	1349	1487	294

5.2 EXPERIMENTS ON HIGHWAY

We further evaluate our method on highway environments (Leurent, 2018). Highway is a collection of environments for autonomous driving and tactical decision-making tasks. We design a task with different source and target environments. A source environment called expressway, and a target environment called city way. The difference between source and target environments is the speed of vehicles. On the expressway, vehicles usually drive faster than on the city way. We initialize the speed for each vehicle using normal distribution, the source and target have different means. Figure 3 shows the highway environment and speed ranges for two domains. In this task, the ego-vehicle is driving on a multilane highway populated with other vehicles. The agent's objective is to reach a high speed while avoiding collisions with neighbouring vehicles. Driving on the right side of the road is also rewarded.

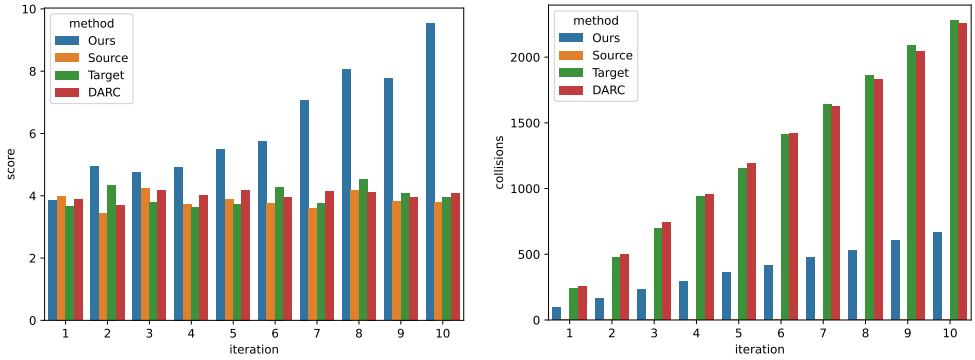
In our experiments, we first learn a policy in the source domain with SAC algorithm (Haarnoja et al., 2018). We interact 100k steps with the source domain and save all transitions as our source offline data. And we collect 10k transitions in the target domain with random policy as initialized target data. During each learning iteration, we additionally allow 1000 interactions with target domain to collect a small amount of new data. Our method collect data with graph-enhanced behaviour policy. Other methods use the current learned policies. We compare our methods with **Source**, **Target** and DARC algorihms as introduced in Section 5.1. And we use AWAC (Nair et al., 2020) as the offline RL algorithm. We use the same network architecture for all algorithms. The network has two hidden layers with 256-units each and ReLU activations. We run each experiment with 5 different random seeds. The performance is evaluated by running 100 episodes at each iteration.



(a) Highway Environment. (b) Speed Ranges.

Figure 3: (a) The illustration of highway environment. (b) The vehicles' speed ranges are initialized with two different distributions in source and target environments.

We show the results in Figure 4. We can find our method achieve the best performance with the least extra interaction. **Source** cannot adapt to the target domain due to unwareness of the dynamics difference. **Target** learns nothing on the target domain due to limited interaction. To our surprise, DARC learn nothing on the target domain, this may because the modified source data conflict with newly collected target data. In addition, from the number of collisions, we can find except for **Source** which does not collect data in the target domain, our method is the safest way to collect data in the target domain. This highlight the effectiveness of graph-enhanced behaviour policy.



(a) The mean score at each iteration. (b) The number of collisions.

Figure 4: (a) The performance of each algorithm on target domain. Each iteration means extra one thousand interactions with the target domain. (b) The number of collisions of each algorithm on target domain.

6 Conclusion

In this work, we aim to solve large dynamics shift problem in offline RL with transfer learning. Our intuition is to make full use of source data to help improve sample efficiency and relax the requirement for the target data. In our method, the source data will play two roles. One is to serve as augmentation data by compensating for the difference in dynamics with modified reward. Another is to form prior knowledge for the behaviour policy to collect a small amount of new data in the target domain safely and efficiently. We learn a policy using offline RL with both the source data and modest amounts of newly collected target data. Experiments on gridworld and autonomous driving environments show that our method achieve high performance with fewer target domain data. And the graph-enhanced behaviour policy collects target data in a safer manner compared with other methods.

REFERENCES

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo Tree Search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Lili Chen, Kimin Lee, Aravind Srinivas, and Pieter Abbeel. Improving computational efficiency in visual reinforcement learning via stored embeddings. *Advances in Neural Information Processing Systems*, 34, 2021.

Benjamin Eysenbach, Shreyas Chaudhari, Swapnil Asawa, Sergey Levine, and Ruslan Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. In *International Conference on Learning Representations*, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062. PMLR, 2019.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *ECML'06 Proceedings of the 17th European conference on Machine Learning*, pp. 282–293, 2006.

Sotiris Kotsiantis and Dimitris Kanellopoulos. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58, 2006.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pp. 45–73. Springer, 2012.

Su Young Lee, Choi Sungik, and Sae-Young Chung. Sample-efficient deep reinforcement learning via episodic backward update. *Advances in Neural Information Processing Systems*, 32, 2019.

Edouard Leurent. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321, 1992.

Jinxin Liu, Hao Shen, Donglin Wang, Yachen Kang, and Qiangxing Tian. Unsupervised domain adaptation with dynamics-aware rewards in reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Jinxin Liu, Zhang Hongyin, and Donglin Wang. Dara: Dynamics-aware reward augmentation in offline reinforcement learning. In *International Conference on Learning Representations*, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

Remo Sasso, Matthia Sabatelli, and Marco A Wiering. Multi-source transfer learning for deep model-based reinforcement learning. *arXiv preprint arXiv:2205.14410*, 2022.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pp. 270–279. Springer, 2018.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*, 2020.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1): 43–76, 2020.

A APPENDIX

A.1 DERIVATION OF REWARD MODIFICATION

Let the optimal policy in the target domain with probability proportional to their exponentiated Q value, where Q is the discounted cumulative rewards. Trajectory τ under the policy in the target domain appear with probability,

$$p(\tau) \propto \rho_0(s_0) \prod_t p_{\text{target}}(s_{t+1}|s_t, a_t) \exp\left(\sum_t \gamma^t r(s_t, a_t)\right), \tag{9}$$

where ρ_0 is the initial state distribution. Similarly, the trajectory in the source domain is,

$$q(\tau) = \rho_0(s_0) \prod_t p_{\text{source}}(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t). \tag{10}$$

Improving the performance of policy $\pi_t heta$ in target domain can achieve by minimizing the reverse KL divergence between $p(\tau)$ and $q(\tau)$,

$$\min_{\pi(a|s)} D_{\text{KL}}(q\|p) = -\mathbb{E}_{p_{\text{source}}}[\sum_t \gamma^t r(s_t, a_t) + \mathcal{H}_\pi[a_t|s_t] + \gamma^t \Delta r(s_t, a_t, s_{t+1})] + c, \tag{11}$$

where $\Delta r(s_t, a_t, s_{t+1}) = \log p_{\text{target}}(s_{t+1}|s_t, a_t) - \log p_{\text{source}}(s_{t+1}|s_t, a_t)$. Using Bayes' rule and the domain classifiers, we have

$$\begin{aligned} p_{\text{target}}(s_{t+1}|s_t, a_t) =& \frac{p(s_{t+1}, s_t, a_t, \text{target})}{p(s_t, a_t, \text{target})} \\ =& \frac{p(\text{target}|s_{t+1}, s_t, a_t)p(s_{t+1}, s_t, a_t)}{p(\text{target}|s_t, a_t)p(s_t, a_t)} \\ =& \frac{p(\text{target}|s_t, a_t, s_{t+1})}{p(\text{target}|s_t, a_t)} + C \end{aligned} \tag{12}$$

Similarly, we have

$$p_{\text{source}}(s_{t+1}|s_t, a_t) = \frac{p(\text{source}|s_t, a_t, s_{t+1})}{p(\text{source}|s_t, a_t)} + C \tag{13}$$

$C = p(s_{t+1}|s_t, a_t)$ is a constant. The learned classifiers $q_{\theta_{\text{SAS}}}(\text{target}|s_t, a_t, s_{t+1})$, $q_{\theta_{\text{SA}}}(\text{target}|s_t, a_t)$ can be good approximations for $p(\text{target}|s_t, a_t, s_{t+1})$ and $p(\text{target}|s_t, a_t)$. So Δr can be written as:

$$\Delta r(s_t, a_t, s_{t+1}) = \log \frac{q_{\theta_{\text{SAS}}}(\text{target}|s_t, a_t, s_{t+1})}{q_{\theta_{\text{SA}}}(\text{target}|s_t, a_t)} - \log \frac{q_{\theta_{\text{SAS}}}(\text{source}|s_t, a_t, s_{t+1})}{q_{\theta_{\text{SA}}}(\text{source}|s_t, a_t)}. \tag{14}$$

And the transition in the source data is modified as $(s_t, a_t, r + \Delta r, s_{t+1})$.