IMITATING EXPERT BEHAVIOUR WITH OPTIMAL TRANSPORT DISTANCES

Neelesh Verma *
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794, USA
{neverma@cs.stonybrook.edu}

ABSTRACT

Offline RL requires the dataset to be reward annotated which can be difficult and labor-intensive. In this report, we are presenting an algorithm for reward annotation in offline RL using the Optimal Transport strategy. We leverage optimal transport to compute an optimal alignment between an unlabeled trajectory and an expert demonstration. The results show that our algorithm outperforms state-of-the-art results in some environments and performs exceptionally well in others. The code is available at GitHub Repo.

1 INTRODUCTION

Offline RL enables learning policies from logged experiences without interaction with the environment. However, offline RL requires a reward function for labeling the logged experience, which can be impractical for applications where rewards are hard to specify. An alternative way to inform the agent about human preference is to provide expert demonstrations, which have been used quite a lot in robotics. One such framework for learning policies from demonstrations is imitation learning (IL), which aims to learn policies that imitate the behavior of expert demonstrations without an explicit reward function.

There are 2 common approaches for IL: Behavioural Cloning (BC), and Inverse Reinforcement Learning (IRL). BC relies on the quality of expert demonstrations. It recovers the expert's behavior by using a supervised learning algorithm. However, BC does generalize well in new situations. On the other hand, IRL learns an immediate reward function and uses that reward. But it works well only on online RL areas, where the agent can have a large number of environment interactions. A high-level overview is summarized in figure 1.

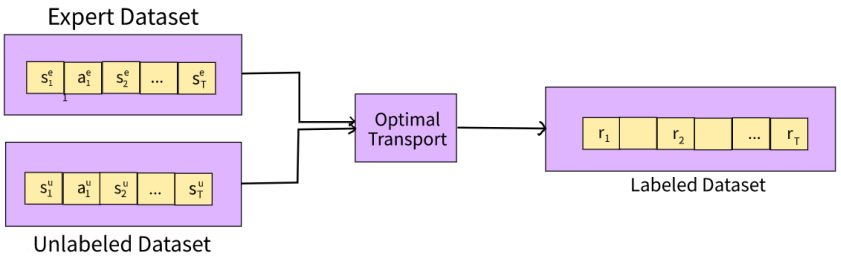

Figure 1: Given expert demonstrations and offline dataset, our algorithm annotates the unlabeled trajectories with rewards which can be fed into an offline RL algorithm

In this project, we have explored the field of Optimal Transport for annotating the trajectories. We will assume that we have at least one expert demonstration available. Using the expert demonstration(s), we find their optimal alignments with unlabeled trajectories in the dataset. The similarity

*Advised by Prof. Michael Ryoo (mryoo@cs.stonybrook.edu) and Xiang Li (xiangli8@cs.stonybrook.edu), Department of Computer Science, Stony Brook University.

measure between a state in an unlabeled trajectory and that of an expert trajectory is treated as a reward label. Once, we have the reward annotated trajectories, we can use any offline RL algorithm.

We have tested our algorithm on many different environments and against algorithms that have ground-truth rewards. Our algorithm achieves consistently better performance (in most environments) as compared to other reward learning approaches.

2 LITERATURE SURVEY

Offline RL aims to learn policies from fixed offline datasets. This technique is helpful when online data collection is expensive and we already have logged experiences available. There has been a good development in the field of reward annotation for offline datasets. Zolna et al. (2020) proposed Offline Reinforced Imitation Learning (ORIL) that learns reward function by contrasting observations from demonstrator and unlabeled trajectories.

Inverse Reinforcement Learning learns a reward function based on an expert dataset. Ho & Ermon (2016) came up with a general framework for directly extracting a policy from data. But it requires a huge number of online samples during training. Moreover, similar to GANs, GAIL also is difficult to optimize and requires very careful tuning of hyper-parameters.

Optimal Transport methods have also been used in IRL approaches. Dadashi et al. (2020) proposed PWIL that minimizes the Wasserstein distance. Using Wasserstein distance directly comes with optimization problems. Although there are approaches to overcome this issue (using dual formulation), they still require a large number of online samples. Our approach addresses all these issues and performs on par with state-of-the-result offline algorithms.

3 METHODOLOGY AND ALGORITHM

3.1 SETTING

We consider a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, p, r, \gamma, p_0, T)$. We have an offline dataset of trajectories (without rewards). Let $\tau = \{s_1, a_1, s_2, a_2, ..., s_T, a_T\}$ denote a single trajectory. We have access to expert demonstrations $\mathcal{D}_{expert} = \{\tau_e^1, \tau_e^2, ..., \tau_e^n\}$. We also have unlabeled offline trajectories $\mathcal{D}_{unlabeled} = \{\tau_u^1, \tau_u^2, ..., \tau_u^m\}$. Our aim is to learn policy π that maximizes the episodic return (assuming MDP is episodic and finite horizon).

3.2 LEVERAGING OPTIMAL TRANSPORT USING WASSERSTEIN DISTANCE

Optimal Transport constructs an alternative notion of distance between probability distribution. One commonly used metric is *Wasserstein Distance*. Let (M, d) be a metric space. For $p \in [1, \infty)$, we define the Wasserstein p-distance between two probability measures μ and ν on M with finite $p-$moments is -

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu, \nu)} \mathbf{E}_{(x,y)\sim\gamma} d(x, y)^p \right)^{1/p} \tag{1}$$

where $\Gamma(\mu, \nu)$ is the set of all couplings of μ and ν. A coupling is a joint measure on $M \times M$ whose marginals are μ and ν on the first and second factors respectively.

Let the state-action distribution of the policy π we seek to obtain be $\hat{\rho}_\pi$ and state-action distribution of the expert be $\hat{\rho}_e$, our objective is to optimize the following problem (Dadashi et al. (2020))-

$$W_p(\hat{\rho}_\pi, \hat{\rho}_e) = \inf_{\pi \in \Pi} \inf_{\theta \in \Theta} \sum_{i=1}^{T} \sum_{j=1}^{D} d((s_i^\pi, a_i^\pi), (s_j^e, a_j^e))^p \theta[i, j] \tag{2}$$

Let's assume the optimal coupling be θ_π^* for the policy π. Then we can define our reward to be -

$$r(s_t^\pi) = -\sum_{j=1}^{D} d((s_t^\pi, a_t^\pi), (s_j^e, a_j^e))\theta[t, j] \tag{3}$$

The reward equation is simple to interpret. The more distance of a trajectory from the expert trajectory, the less reward it gets. Please note that we have only considered the case of $p = 1$. For $p \geq 1$, we leave this for future cases (future work).

3.3 Our Algorithm

Now, we have defined how to compute the reward function, we can write our final algorithm as follows. Before that, we will assume that we have an efficient way of solving equation 2 (we will mention the packages and frameworks in the Experiments section).

Algorithm 1 Pseudo-Code for Reward Labelling with Optimal Transport

Input : Unlabeled Dataset $\mathcal{D}_{unlabeled}$ and, Expert Dataset $\mathcal{D}_{expert}$
Output : Reward annotated dataset $D_{labeled}$

$\mathcal{D}_{labeled} \leftarrow \phi$
for τ_i in $\mathcal{D}_{unlabeled}$ **do**
  $d_i, \theta^{*(i)} \leftarrow$ OptimalTransport$(\mathcal{D}_{expert}, \tau_i)$
  for each time step t **do**
    $r(s_t^i) = -\sum_{j=1}^{D} d_{t,j}^i \theta_{t,j}^{*(j)}$
  end for
  $\tau = (s_1^i, a_1^i, r(s_1^i), ..., s_T^i, a_T^i, r(s_T^i))$
  $\mathcal{D}_{labeled} = \mathcal{D}_{labeled} \cup \tau$
end for

In the algorithm 1, we start with our unlabeled dataset and expert dataset. For each of the trajectories in the unlabeled dataset, we solve the Optimal transport problem (optimal alignment between expert trajectories and unlabeled trajectories). After solving the OT problem, we have our distance metric d and optimal coupling θ^*. Then using equation 3, we compute the reward for each state in the unlabeled trajectory. Once we have the reward for each state, we can annotate our trajectory and put it in a new dataset $\mathcal{D}_{labeled}$. This way, after the completion of the algorithm, we will have a new dataset with reward annotations. In the case of having multiple expert trajectories, we will compute the episodic rewards from each of the expert trajectories and use the rewards from that expert trajectory that gives the best episodic return.

We can use any underlying offline RL algorithm with this new dataset. For all our experiments and results, we have used the standard offline algorithm - *Implicit Q-Learning* proposed by Kostrikov et al. (2021).

4 Experiments and Results

In this section, we will look at - Implementation details, Environments used, Baselines, and Results.

4.1 Implementation Details

Our speed of the algorithm is very much dependent on how we solve the Optimal transport problem. Chizat et al. (2020) proposed a faster way to estimate the Wasserstein distance using Sinkhorn divergence. We are not going to discuss how Sinkhorn achieves this (please read their paper). However, there wasn't any implementation of Sinkhorn in PyTorch (there was one but it didn't work for me). Instead, we have used a JAX-based implementation Cuturi et al. (2022). The underlying algorithm is IQL (Imitation Q-Learning), however, any other algorithm can also be used. For the distance function, we have used cosine distance as proposed by many papers.

We have run our code on a 4-cluster GPU (two 6 GBs GTX 1080, and two 8 GBs GTX TITAN Z). JAX is able to take advantage of parallelization across GPUs.

4.2 Environments and Datasets

We needed both an expert dataset and an unlabeled dataset. We have used D4RL datasets Fu et al. (2020). We have run our algorithm on two different tasks - Gym MuJoCo Locomotion tasks (*HalfCheetah-v2*, *Hopper-v2* and *Walker-v2*), and AntMaze (*antmaze-v0*). We used the *medium* and *medium-expert* datasets for each MuJoCo task to get our expert dataset and unlabeled dataset. For antmaze, we used *umaze* and *medium* datasets. To build the expert dataset, we simply chose the episodes having the best episodic return. Since the dataset contains the reward information, we simply discarded that for our unlabeled dataset.

4.3 Baselines

We have used the following baselines offline RL algorithms -

- IQL - This is our underlying offline RL algorithm (Kostrikov et al. (2021)). We showed in the results how our reward annotations improve the performance of IQL.
- ORIL - It also learns reward function (Zolna et al. (2020)), so we can compare our reward learning with it. We have used IQL as the underlying algorithm for this as well.
- DemoDICE - This is an offline Imitation learning algorithm (Kim et al. (2022)) that is the current state-of-the-art.

4.4 Results

Table 1: Performance on different Gym-MujoCo Locomotion Tasks

Dataset	IQL	IQL + ORIL	DemoDice	IQL + Ours
halfcheetah-medium-v2	46.8	**48.4**	42.9	43.1
halfcheetah-medium-expert-v2	85.7	87.1	85.3	**88.9**
hopper-medium-v2	65.9	46.7	54.8	**72.2**
hopper-medium-expert-v2	91.2	29.5	**92.7**	**92.5**
walker-medium-v2	77.8	61.5	72.9	**77.4**
walker-medium-expert-v2	109.1	**110.4**	105.8	108.7

We started by using only a single expert demonstration. For the Gym-MuJoCo locomotion tasks, we can see in table 1 that our algorithm (combined with IQL) outperforms the baseline in 3 tasks. We have also experimented with a different task - *antmaze-v0* on *umaze* and *medium datasets.* The results are shown in table 4.4 -

Dataset	IQL	IQL + Ours
antmaze-medium-v0	74.8	73.2
antmaze-umaze-v0	88.2	85.4

Table 2: Performance against IQL having ground-truth rewards

Our algorithm is able to (almost) recover the performance of the underlying IQL algorithm having ground-truth rewards. Some more experiments are needed to be done to establish this.

5 Novelty

There have been optimal transport approaches (Dadashi et al. (2020)) but they were for Online settings requiring a large number of environmental interactions. We have presented a new approach for offline settings. Moreover, we have used the Sinkhorn method for Wasserstein estimation instead

of using the primal form. We have also used a different cost function as compared to PWIL. The primary reason was to make it more robust to noise. A detailed comparison is given in the appendix.

6 CONCLUSION

We have presented an optimal transport method to annotate the rewards in unlabeled trajectories using an expert dataset. We can observe that our optimal transport strategy for offline settings is able to recover the performances of offline RL algorithms having access to ground-truth rewards. Moreover, it outperforms earlier reward annotation algorithms and imitation learning algorithms (ORIL, DemoDice). Additionally, our strategy can be combined with any other underlying offline RL algorithms (we only used IQL). This method is very suitable for cases when online data collection is expensive or not feasible.

7 FUTURE WORK

Considering the Wasserstein equation 2, we have only explored the case of $p = 1$. For higher values of p, the effect on the performance needs to be studied. It is also interesting to see the overall effect, when we don't have expert trajectories (when we only have some decent episodic returns trajectories instead of maximum return ones). So, a possible extension of this algorithm could be the case when expert demonstration collection is also difficult.

REFERENCES

Lenaic Chizat, Pierre Roussillon, Flavien Léger, François-Xavier Vialard, and Gabriel Peyré. Faster wasserstein distance estimation with the sinkhorn divergence. *Advances in Neural Information Processing Systems*, 33:2257–2269, 2020.

Marco Cuturi, Laetitia Meng-Papaxanthos, Yingtao Tian, Charlotte Bunne, Geoff Davis, and Olivier Teboul. Optimal transport tools (OTT): A JAX toolbox for all things wasserstein. *CoRR*, abs/2201.12324, 2022. URL https://arxiv.org/abs/2201.12324.

Robert Dadashi, Léonard Hussenot, Matthieu Geist, and Olivier Pietquin. Primal wasserstein imitation learning. *CoRR*, abs/2006.04678, 2020. URL https://arxiv.org/abs/2006.04678.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: datasets for deep data-driven reinforcement learning. *CoRR*, abs/2004.07219, 2020. URL https://arxiv.org/abs/2004.07219.

Scott Fujimoto and Shixiang Gu. A minimalist approach to offline reinforcement learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Q32U7dzWXpc.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *CoRR*, abs/1606.03476, 2016. URL http://arxiv.org/abs/1606.03476.

Geon-Hyeong Kim, Seokin Seo, Jongmin Lee, Wonseok Jeon, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. DemoDICE: Offline imitation learning with supplementary imperfect demonstrations. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=BrPdX1bDZkQ.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *CoRR*, abs/2110.06169, 2021. URL https://arxiv.org/abs/2110.06169.

Konrad Zolna, Alexander Novikov, Ksenia Konyushkova, Çaglar Gülçehre, Ziyu Wang, Yusuf Aytar, Misha Denil, Nando de Freitas, and Scott E. Reed. Offline learning from demonstrations and unlabeled experience. *CoRR*, abs/2011.13885, 2020. URL https://arxiv.org/abs/2011.13885.

A APPENDIX

A.1 SETTINGS OF HYPER-PARAMETERS

For the underlying IQL, we have used the same set of hyper-parameters as suggested in the original paper (Kostrikov et al. (2021)). Also, in the implementation of PWIL by Dadashi et al. (2020), the reward computed in the equation 3 is further exponentiated to make it more sensitive to small differences in cost. The exponential function is a monotonically increasing function, which means that it always increases as the input increases. This means that even small differences in cost will lead to large differences in reward.

Additionally, exponentiating the cost makes the reward function more robust to noise. This is because noise can cause small changes in the cost, but these changes will be amplified by the exponential function. This means that the reward function will still be able to distinguish between states and actions that are similar, even if there is some noise in the data. The new reward function is -

$$r_{new} = \alpha \exp(\beta r_{old})$$

where r_{old} is the one computed in equation 3, α and β are the parameters controlling the range of reward. The list of hyper-parameters (for MuJoCo Locomotion tasks) is summarized in table A.1 -

Hyper-parameter	Value
Episode Length	1000
Distance Function (d)	Cosine
α	5

Table 3: Hyper-parameters for Gym MuJoCo Locomotion Task

The value of α is the same for MuJoCo and Antmaze, but the values of β are different. Please refer to Dadashi et al. (2020) for their setting.

A.2 VARYING NUMBER OF EXPERT DEMONSTRATIONS

So far, we have only used 1 expert demonstration in all our results. We observed that on increasing the number of expert demonstrations to 10, our results became even better (in most cases). The results are summarized in the table A.2 -

Dataset	Expert Demo = 1	Expert Demo = 10
halfcheetah-medium-v2	43.1	43.8
halfcheetah-medium-expert-v2	88.9	89.1
hopper-medium-v2	72.2	72.1
hopper-medium-expert-v2	92.5	93.2
walker-medium-v2	77.4	78.5
walker-medium-expert-v2	108.7	109.4

Table 4: Varying number of Expert Demonstrations for MuJoCo

A similar trend was found in the Antmaze environment. The results are shown in table A.2.

Dataset	Expert Demo = 1	Expert Demo = 10
antmaze-medium-v0	73.2	75.6
antmaze-umaze-v0	85.4	90.7

Table 5: Varying number of Expert Demonstrations for Antmaze Datasets

A.3 VARYING HYPER-PARAMETERS

We tested varying two hyper-parameters α and β. The original values (=5) were proposed in the PWIL paper. The results are summarized in table A.3. We can see that the original parameters are indeed the best-performing ones.

Dataset	$\alpha = \beta = 5$	$\alpha = \beta = 1$
halfcheetah-medium-v2	43.8	42.7
halfcheetah-medium-expert-v2	89.1	87.1
hopper-medium-v2	72.1	75.3
hopper-medium-expert-v2	93.2	89.4
walker-medium-v2	78.5	77.6
walker-medium-expert-v2	109.4	107.2

Table 6: Varying α and β (using 10 expert demonstrations)

A.4 DIFFERENT UNDERLYING ALGORITHM THAN IQL

Lastly, we wanted to see the effect on performance if we use some other underlying algorithm than IQL. We experimented with another state-of-the-art offline RL algorithm TD3+BC proposed by Fujimoto & Gu (2021). The results (using single expert demonstrations) are summarized in table A.4.

Dataset	TD3+BC	TD3+BC+Outs
halfcheetah-medium-v2	47.8	43.1
halfcheetah-medium-expert-v2	92.8	88.3
hopper-medium-v2	66.4	71.1
hopper-medium-expert-v2	106.4	108.5
walker-medium-v2	84.2	78.5
walker-medium-expert-v2	108.4	108.5

Table 7: Using TD3+BC as underlying algorithm

We can observe that the performance of TD3+BC using our optimal transport strategy mostly matches the original algorithm using ground-truth rewards.