Stable Control Policy and Transferable Reward Function via Inverse Reinforcement Learning

Keyu, wu
Institute of Software Chinese Academy of Sciences,
University of Chinese Academy of Sciences

FENGGE, Wu
Institute of Software Chinese Academy of Sciences,
University of Chinese Academy of Sciences

YIJUN, Lin
Institute of Software Chinese Academy of Sciences,
University of Chinese Academy of Sciences

JUNSUO, Zhao
Institute of Software Chinese Academy of Sciences,
University of Chinese Academy of Sciences

ABSTRACT

Inverse reinforcement learning (IRL) can solve the problem of complex reward function shaping by learning from expert data. However, it is challenging to train when the expert data is insufficient, and its stability is difficult to guarantee. Moreover, the reward function of mainstream IRL can only adapt to subtle environmental changes. It cannot be directly transferred to a similar task scenario, so the generalization ability still needs to be improved. To address these issues, we propose an IRL algorithm to obtain a stable control policy and transferable reward function (ST-IRL). Firstly, by introducing the Wasserstein metric and adversarial training, we solve the problem that IRL is challenging to train in a new environment with little expert data. Secondly, we add state marginal matching (SMM), hyperparameter comparison and optimizer evaluation to address the model's generalisability problem. As a result, the control policy obtained by ST-IRL achieves outstanding control results in all four Mujoco benchmarks. Furthermore, in both the custom Ant and PointMaze environments, the reward function obtained by our algorithm exhibits promising transferability.

CCS CONCEPTS

• **Computing methodologies**; • **Policy iteration**;

KEYWORDS

Inverse reinforcement learning, Reward function, Wasserstein metric, Adversarial training

ACM Reference Format:
Keyu, wu, FENGGE, Wu, YIJUN, Lin, and JUNSUO, Zhao. 2023. Stable Control Policy and Transferable Reward Function via Inverse Reinforcement Learning. In *2023 9th International Conference on Computing and Artificial Intelligence (ICCAI 2023), March 17–20, 2023, Tianjin, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3594315.3594399

1 INTRODUCTION

With a series of breakthroughs in the field of games [1–3], and robotics [4], and with successful real-world applications [5–7], the research community has seen a surge of interest in reinforcement learning (RL) algorithms [8]. Although RL has impressive achievements, many implementations are drawn from specific cases and cannot be transferred to similar domains. The main reason is that when RL is used in an uncertain environment, the first challenge is modeling the environment. Researchers rely on environmental cognition and engineering experience to complete the shaping of the reward function. However, it is highly stochastic and complex for the data, scenarios, knowledge domains and other elements of the environment [9] to deeply interact with the design or operation of the system. As a result, describing the reward function formally in a new environment is quite challenging.

Imitation learning (IL) [10] and IRL [11, 12] have laid a good foundation for solving the problem that the environment is difficult to be described accurately. IL attempts a direct learning policy but relies on a stable environment to remain optimal. If the environment changes, IL will require expert data from the new environment to learn again. To some extent, IRL can solve the problem of dynamic environment changes. The reward function learned by IRL is independent of the dynamics of the environment, and it is an invisible map of the task objective. Therefore, the policy will have better universality by relearning the reward function obtained by IRL.

However, it is significantly difficult for IL and IRL to obtain expert data in a new environment. In most scenarios, expert data is inadequate, and is in a high-dimensional continuous space. In such a context, IL faces the covariate shift problem [13]. Moreover, IRL suffers from high training difficulty, non-convergence of control policy and poor generalization of reward functions.

After analyzing the above problem, we find that the mainstream IRL [14–19] measures the distribution between the expert trajectories and the policy trajectories through f-divergence [20]. When the data is in a high-dimensional space and the amount of data is small, the results of the metric tend to be constant or infinite. This situation is reflected in the training process that the model does not converge or the convergence speed is slow.

Therefore, we focus on the metric functions and combine it with the idea of adversarial training to propose ST-IRL. In order to get a stable control policy and facilitate comparison with other algorithms, We use the Soft Actor-Critic (SAC) model [21] to construct our control policy. Likewise, to get a transferable reward function, we use the Multilayer Perceptron (MLP) to build a transferable reward function. Furthermore, we use adversarial training to iterate the control policy and the reward function to increase the robustness between models. The divergence in the reward function

This work is licensed under a Creative Commons Attribution International 4.0 License.

ICCAI 2023, March 17–20, 2023, Tianjin, China
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9902-9/23/03.
https://doi.org/10.1145/3594315.3594399

is measured by Wasserstein metric [22], which solves the problem of non-convergence or slow convergence of the model caused by the small amount of data in the high-dimensional space. Hyperparameters and optimizer screening carry out the training process. Finally, the core idea in SMM is adopted to improve the model's generalization ability by focusing on state data distribution.

The main contributions of this work are as follws:

- We propose an inverse reinforcement learning algorithm, ST-IRL, by which we can obtain a control policy with better performance and stability.
- With ST-IRL, we can obtain a reward function with transferability.

2 RELATED WORK

Two branches of research inspired the ST-IRL: one started from the perspective of the metric function, minimizing the distribution of policy data and expert data. The other introduced the idea of Generative Adversarial Network (GAN), using the reward function as a discriminator for training to obtain the optimal reward function.

The first branch proceeded from the metric function. MaxEntIRL [14] determined the policy and reward function by minimizing the forward Kullback-Leibler (KL) divergence among policy trajectories based on the maximum entropy. Similar to MaxEntIRL, ReEntIRL [23], Deep MaxEntIRL [15] and GCL [16] optimized the forward KL divergences among policy trajectories. However, there lay some differences. ReEntIRL modeled the task as a relative entropy problem, while Deep MaxEntIRL and GCL estimated the reward function by constructing a deep neural network in reward functions. Starting with reverse KL, SMM [24] and EBIL [25] minimized the difference between expert trajectories and policy trajectories. Finally, f-IRL [26] aggregated f-divergence to achieve IRL optimization on forward KL, reverse KL and Jensen-Shannon (JS) metrics.

The second branch brought in the GAN model. Finn et al. [27] first proposed the combination of IRL and GAN in 2016, and many algorithms were further studied following this direction. GAIL [17] and f-MAX [18] picked expert data from state-action pairs, using the policy model as a generator to produce actions with the state data as input. At the same time, the reward function acted as a discriminator to judge the similarity between the generated action and the expert action. AIRL [19] decoupled the reward function into a state-only function, which was more robust than GAIL.

We extend the above ideas from f-divergence to the Wasserstein metric. Using the Wasserstein metric, ST-IRL can successfully measure the divergence among trajectory distributions when the trajectory data is in a high-dimensional space and the amount of data is small. Thus the problem of gradient disappearance or gradient explosion can be solved. We borrow the idea of combining IRL and adversarial training, which uses the control policy as the generator and the reward function as the discriminator for adversarial training. The differences from previous IRL algorithms are:

- We evaluate two different implementations of the Wasserstein metric and propose the optimal IRL based on the Wasserstein metric.
- We construct the reward function through MLP, decouple it in adversarial training, and generalize the trained reward function to similar tasks.

3 METHOD

This section focuses on the ST-IRL algorithm and analyzes why the ST-IRL is more stable in high-dimensional spaces with little expert data.

3.1 Preliminaries

In this section, we review the definitions of the Wasserstein metric, the state marginal matching (SMM) and the generative adversarial network (GAN) that we build upon in this work.

3.1.1 Wasserstein Metric. The definition of the Wasserstein metric is shown in equation (1).

$$W\left(P_e, P_a\right) = \inf_{\gamma \sim \Gamma} \mathbb{E}_{(x,y)\sim\gamma}\left[x - y\right] \tag{1}$$

where $P_e(P_{expert})$ and $P_a(P_{agent})$ are two distributions over X. X is a compact metric set (such as the space of images $[0,\ 1]^d$). Γ is the set of all possible joints on $X \times X$ that have marginals P_e and P_a. For each possible joint distribution γ, a sample x and y can be obtained by sampling $(x,\ y) \sim \gamma$ from it and calculating the distance $x - y$ for this pair of samples. Intuitively, γ indicates how much "mass" must be transported from x to y in order to transform the distributions P_e into the distribution P_a. The lower bound $\mathbb{E}_{(x,y)\sim\gamma}[x - y]$ that can be taken on this expectation across all possible joint distributions is the Wasserstein metric.

3.1.2 State Marginal Matching. The state marginal distribution $P(s)$ is a distribution of states, not trajectories: It is the distribution over states visited in a finite-length episode, not the stationary distribution of the policy after infinitely many steps. The definition of the state marginal distribution $P(s)$ is shown in equation (2).

$$P\left(s\right) \triangleq \mathbb{E}_{\substack{s_1 \sim p_0\left(S\right) \\ a_t \sim \pi\left(A|s_t\right) \\ s_{t+1} \sim p\left(S|s_t,\ a_t\right)}} \left[\frac{1}{T}\sum_{t}^{T}\mathbb{F}\left(s_t = s\right)\right] \tag{2}$$

where S is a state set and A is the action set in a Markov Decision Process (MDP) with fixed episode lengths T, s_1 is an initial state sampled from the initial state distribution $p_0(S)$, π is a parametric policy, $p(S|s_t, a_t)$ is a dynamics distribution, $\sum_{t}^{T}\mathbb{F}(s_t = s)$ is the visitation count of a state s in a finite-length episode. Given the expert state distribution $P_e(s)$, we can train an agent to match the expert behavior by minimizing the following Wasserstein metric objective L_w:

$$L_w\left(\theta\right) = D_w(P_e\left(s\right)||P_a\left(s\right)) \tag{3}$$

where D_w is the definition of the Wasserstein metric mentioned before in equation (1).

Our algorithm will compute the analytical gradient of equation (3) w.r.t. θ and optimize the reward function via gradient descent.

3.1.3 Generative Adversarial Network. The first step of the GAN is to fix the generator and update the discriminator, where the loss function of the discriminator is equation (4).

$$-\mathbb{E}_{x\sim P_e(s)}\left[logD\left(x\right)\right] - \mathbb{E}_{x\sim P_a(s)}\left[\log\left(1 - D\left(x\right)\right)\right] \tag{4}$$

where $P_e(s)$ is an expert state distribution, $P_a(s)$ is the agent state distribution, and $D(x)$ is the discriminator.

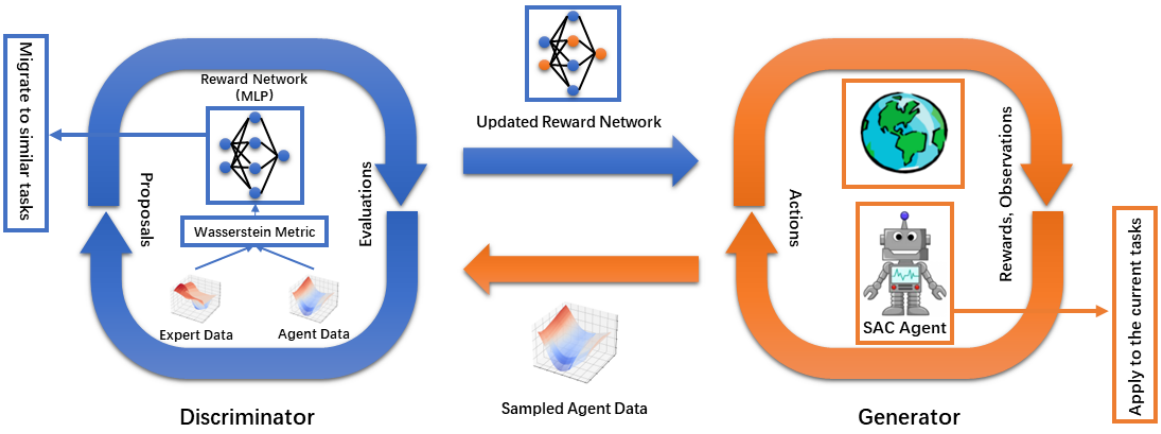

Figure 1: Model structure diagram. The yellow part is the generator, and the blue part is the discriminator.

The second step is to fix the discriminator and update the generator, where the loss function of the generator is equation (5).

$$\mathbb{E}_{x\sim P_a(s)}\left[\log\left(1-D\left(x\right)\right)\right] \tag{5}$$

3.2 Model structure

The model structure of ST-IRL can be divided into a generator and a discriminator. The model uses adversarial training to iterate, in order to achieve the Nash equilibrium between the generator and the discriminator. The overall training process of ST-IRL is as follows: Firstly, the generator samples the state data $P_a(s)$ of the trained SAC agent and inputs it into the discriminator D. And then, the discriminator D updates the reward function r_θ by minimizing the divergence between the expert state distribution $P_e(s)$ and agent state distribution $P_a(s)$ (see Algorithm 1). Finally, the updated reward function r'_θ is used as the environment evaluation function for the next generator update.

As for the specific structure, inside the generator, the SAC agent interacts with the environment and obtains as many rewards as possible from the environment through continuous iteration. The discriminator uses the Wasserstein metric internally to measure the divergence among the state data distributions and then updates it by gradient descent. After both the SAC agent and the reward network are trained, we apply the SAC agent to the current task to evaluate its stability and feasibility, and then transfer the reward network to similar task scenarios to evaluate its transferability.

Algorithm 1 ST-IRL

require: Expert state distribution $P_e(s)$
ensure: Learned reward function r_θ, Policy π_θ
initialize r_θ, and Discriminator D
for $i \Leftarrow 1$ to $Iter$ do
  $\pi_\theta \Leftarrow$ ST-IRL(r_θ) and collect agent trajectories T_θ
  fit the discriminator D by equation (4) using expert and agent state samples from $P_e(s)$ and T_θ
  compute sample gradient $\nabla_\theta L_w(\theta)$ over T_θ
  $\theta \Leftarrow \theta - \lambda\nabla_\theta L_w(\theta)$
end

3.3 Principle analysis

In Fig. 1, the state data distribution $P_a(s)$ is generated depending on the interaction between the agent in the generator and the environment. If the data distribution in this model is in a high-dimensional space, it satisfies the definition of a low-dimensional manifold in a high-dimensional space [28]. The policy generally maps low-dimensional discrete state data into high-dimensional continuous action data via a deep neural network. If the agent is fixed, the distribution of the data will be limited by the low-dimensional space, although the sampled data is defined in the high-dimensional space. So the final support set of the data distribution becomes a low-dimensional flow in the high-dimensional space. When the amount of expert data is small, the expert data cannot fill the entire high-dimensional space. Therefore, the expert state data distribution $P_e(s)$ and the sampling state data distribution $P_a(s)$ obtained by the agent are in the state of no overlap or negligible overlap.

The mainstream IRL updates the control policy and reward function by measuring the divergence between $P_e(s)$ and $P_a(s)$, using the f-divergence. When there is no overlap or negligible overlap between the two distributions, the f-divergence measure is either infinite or constant (refer equation (6)). This means that the gradient descent update method is not suitable.

$$\begin{aligned} RKL(P_e\left(s\right)||P_a\left(s\right)) &= +\infty \\ KL(P_e\left(s\right)||P_a\left(s\right)) &= +\infty \\ JS(P_e\left(s\right)||P_a\left(s\right)) &= log2 \end{aligned} \tag{6}$$

where RKL is reverse Kullback-Leibler divergence, KL is Kullback-Leibler divergence, and JS is Jensen-Shannon (JS) metrics.

The advantage of the Wasserstein metric is that even if there is no overlap between two distributions, it still reflects the divergence between the distributions (see Appendix A).

However, the definition of Wasserstein metric cannot be solved directly in the discriminator model, so we resort to the idea of Lipschitz GAN [29] to construct it in the form of a loss function by

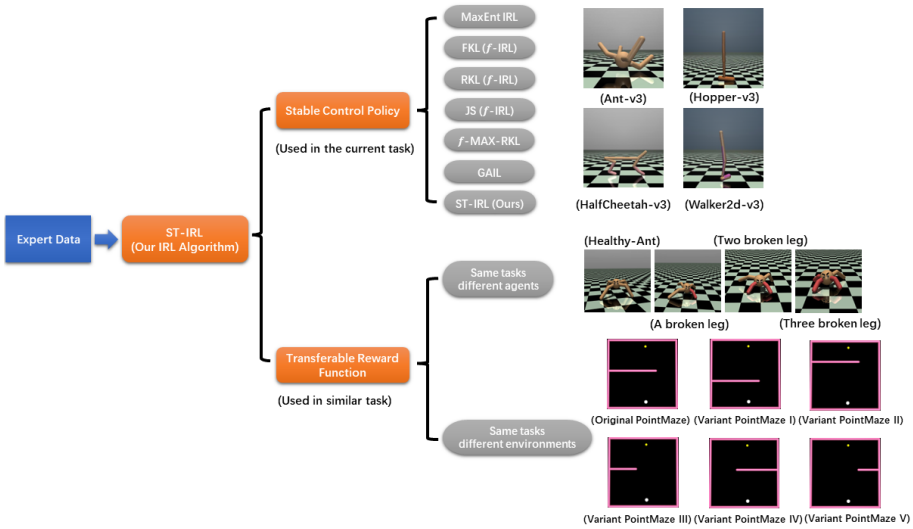

Figure 2: Experimental outline. The diagrams in the top-right corner of the illustration above show the four classic Mujoco environments. The diagrams in the middle show the custom Ant environment, and the diagrams on the bottom show six custom PointMaze environments. PointMaze is a square with length=1. The most significant difference in each PointMaze environment is the position and length of the middle bar. For example, the original PointMaze has the bar on the left, with length=2/3 and height=1/2. The variant PointMaze I has the bar on the left, with length=2/3 and height=1/3. The variant PointMaze II has the bar on the left, with length=2/3 and height=2/3. The variant PointMaze III has the bar on the left, with length=1/3 and height=1/2. The variant PointMaze IV is on the right, with length=2/3 and height=1/2. The variant PointMaze V is on the right, with length=1/3 and height=1/2.

incorporating Lipschitz continuity (refer equation (7)).

$$W\left(P_e\left(s\right), P_a\left(s\right)\right) = \frac{1}{K} \underset{f_L \leq K}{sup} \mathbb{E}_{x \sim P_e(s)}\left[f\left(x\right)\right] - \mathbb{E}_{x \sim P_a(s)}\left[f\left(x\right)\right] \tag{7}$$

where the supermum is over all the K-Lipschitz functions. The K-Lipschitz function actually imposes an additional restriction on top of a continuous function f by requiring the existence of a constant $K \geq 0$. So any two elements x_1 and x_2 in the domain of definition satisfy $|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$.

4 EXPERIMENTS

The experimental section focuses on answering two questions (see Fig. 2):

- How effective and stable does ST-IRL obtain the control policy?
- How transferable is the reward function obtained via ST-IRL?

To answer the first question, we conduct two pre-experiments. The first pre-experiment evaluates the effect of different Wasserstein metric implementations on the ST-IRL algorithm, and the second pre-experiment evaluates the effect of hyperparameters and optimizers on the ST-IRL. And then, to verify the control effect and stability of ST-IRL, we compare the ST-IRL algorithm with six mainstream IRL algorithms in four classical Mujoco environments. The four classic Mujoco environments are Ant-v3, Hopper-v3, HalfCheetah-v3, and Walker2d-v3, which can well represent the control problems of continuous state space and action space.

To answer the second question, we also perform a pre-experiment to verify whether or not ST-IRL can learn valid information from expert data containing task goals. Finally, in the formal experiments, we test the transferability of ST-IRL's reward function in two scenarios: one with the same task and different agents and the other with the same task and different environments.

The expert state data in the above experiments are all sampled from a stable SAC model.

4.1 Stability verification of control policy

4.1.1 Pre-experiment 1-1. In ST-IRL, we implement two forms of weight clipping [22] and gradient penalty [30] of the Wasserstein metric and obtain ST-IRL-WC and ST-IRL-GP, respectively.

We select expert state data with a trajectory number of 4 to conduct experiments in four Mujoco environments and compare the cumulative sum of the final rewards obtained by the policy. The results (see Fig. 3) show that ST-IRL-WC significantly outperforms ST-IRL-GP in terms of the final cumulative reward and convergence speed. In the two implementations, weight clipping is limited by clipping the network weight w. In contrast, the gradient penalty is limited by setting an additional loss term. The weight clipping approach is valid globally for the whole sample space. However, the gradient penalty is valid only for the proper and false sample concentration regions and the transitional zones in between. The experiment demonstrates that ST-IRL-WC has faster convergence speed and better stability. The ST-IRL algorithms mentioned in the following experiments use weight clipping.

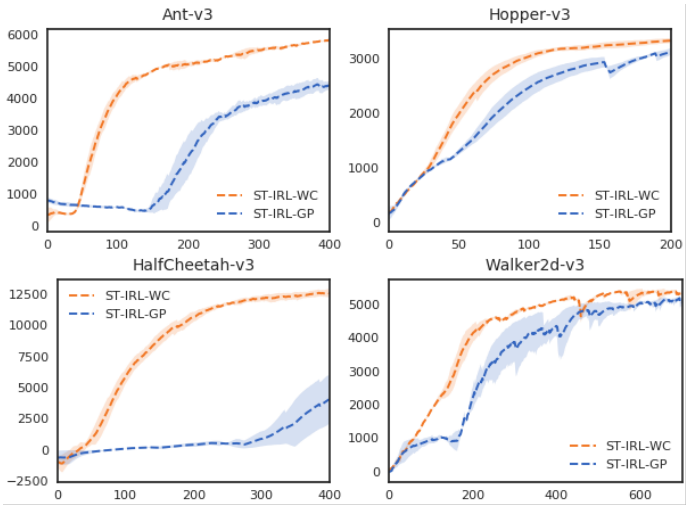

Figure 3: The cumulative return of ST-IRL-WC and ST-IRL-GP in four Mujoco environments.

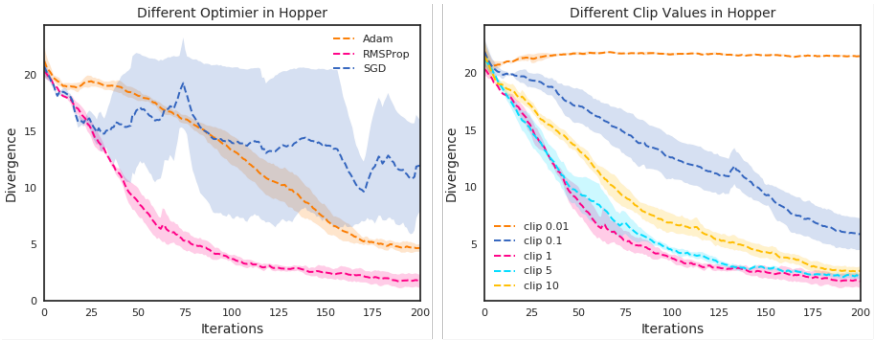

Figure 4: The figure on the left shows the ST-IRL divergence under different optimizers. The right figure shows the ST-IRL divergence in different clip values. All experiments are carried out under Hopper-v3.

4.1.2 Pre-experiment 1-2. This experiment mainly evaluates the impact of different clip values in weight clipping and different optimizers in the model on the ST-IRL algorithm.

As for the clip value, we compare the cases of 10, 5, 1, 0.1, and 0.01. As for the optimizer, we select three optimizers: RMRProp, Adam, and SGD. In the environment of expert trajectories=4 and hopper-v3, we analyze the divergence between the distribution of expert trajectories and the distribution of policy trajectories with different clip values and optimizers. From the experimental results of the optimizer, RMSProp converges faster and is more stable. However, the optimizer of SGD fluctuates a lot, and the optimizer of Adam converges slowly. As for the experimental results of clip values, the optimal results are achieved when clip=1. If the clip value is too small, there will be redundant network clipping, and useful information is hard to learn. However, if the clip value is too large, the convergence speed of the model will be reduced.

4.1.3 Experiment 1. In this section, we answer Question 1 in tabular and graphical form. In four classical Mujoco environments, we compare the ST-IRL algorithm with six mainstream IRL algorithms. Furthermore, we illustrate the training policy's cumulative rewards when the number of expert trajectories is 1, 4 and 16. The cumulative reward is an intuitive reflection of the effectiveness of the policy. The table (refer Table 1 shows the ratio of the policy trajectory's cumulative reward to the expert trajectory's cumulative reward. We trim the ratio directly to 1 when the cumulative reward of the policy trajectory is greater than the cumulative reward of the expert trajectory. The ratios in the table show that ST-IRL can exceed 0.9 in all four experiments, which has strong stability. In addition, ST-IRL can achieve state-of-the-art under HalfCheetah-v3 and Walker2d-v3.

The trend of the curves in Fig. 5 shows that ST-IRL has significantly better convergence and stability than all the algorithms except f-MAX-IRL and GAIL. After analyzing f-MAX-IRL and GAIL, we find that the reward function is tightly coupled with the control policy in the model structure. Instead of generating an independent reward function, a temporary evaluation function is used in the training process to complete the policy training. Such algorithms can achieve good results in policy evaluation. However, they are susceptible to hyperparameters. Besides, they require many tuning parameters in each environment to reach optimality.

Table 1: We compare the performance of seven IRL algorithms in four Mujoco environments, where the bolded values represent the best performance of each experimental environment.

	Ant-v3			Hopper-v3			HalfCheetah-v3			Walker2d-v3		
Expert return	6050			3612			12914			5429		
Expert traj	1	4	16	1	4	16	1	4	16	1	4	16
MaxEnt IRL	0.91	0.96	0.92	0.64	0.87	0.65	0.99	0.83	0.97	**1**	0.99	0.99
FKL(f-IRL)	0.91	0.88	0.91	0.54	0.71	0.80	0.95	0.93	0.99	0.92	0.91	0.95
RKL(f-IRL)	0.94	0.97	0.96	0.53	0.67	0.66	0.98	0.98	0.97	0.98	0.96	0.94
JS(f-IRL)	0.91	0.99	0.90	0.52	0.57	0.68	0.96	0.99	0.96	**1**	**1**	0.97
f-MAX-IRL	**0.99**	**1**	**0.99**	**0.95**	**0.94**	**0.95**	0.72	0.75	0.71	0.59	0.61	0.67
GAIL	0.98	0.98	0.97	0.88	0.91	0.90	0.69	0.40	0.69	0.57	0.58	0.61
ST-IRL	0.93	0.96	**0.99**	0.92	0.93	0.93	**1**	**1**	**0.99**	**1**	**1**	**1**

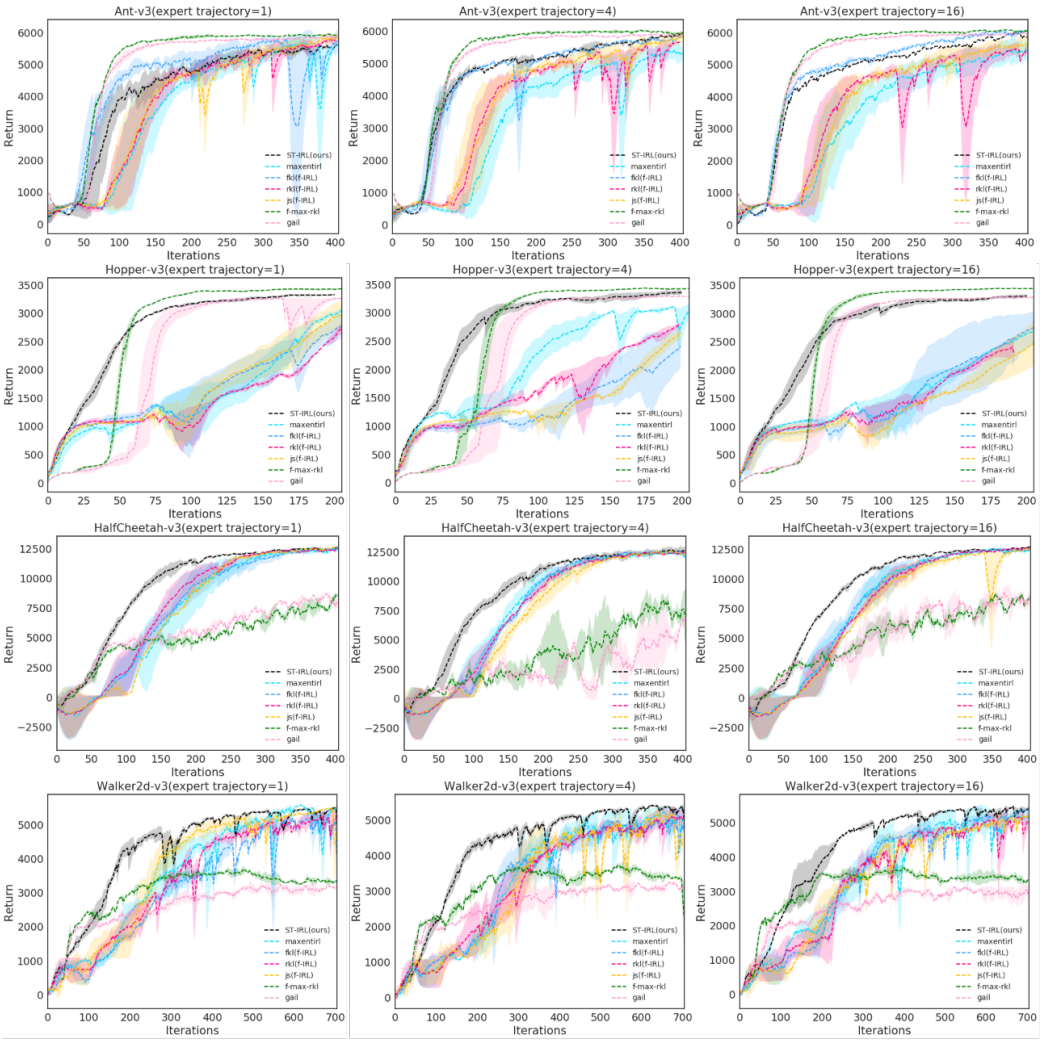

Figure 5: We compare the cumulative returns of seven IRL algorithms for policies in four Mujoco environments with expert trajectory numbers of 1, 4, and 16.

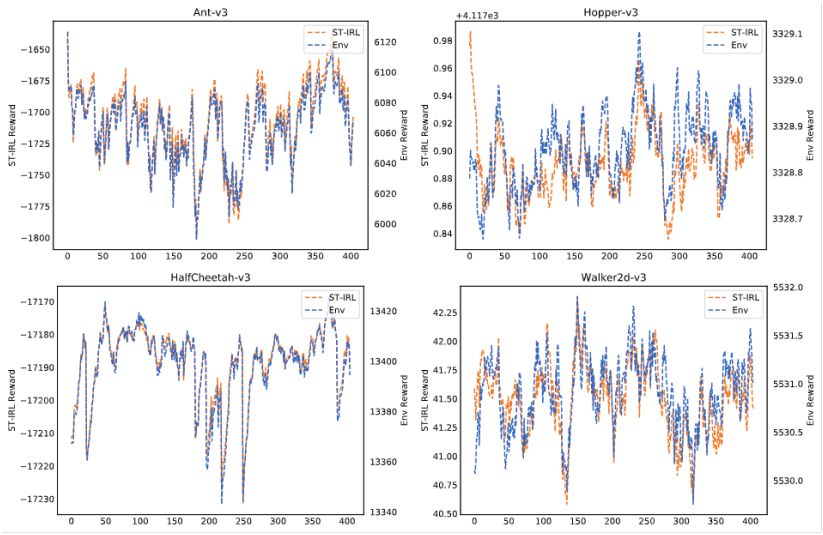

Figure 6: The reward function of ST-IRL and the reward function set in Mujoco score the same random action.

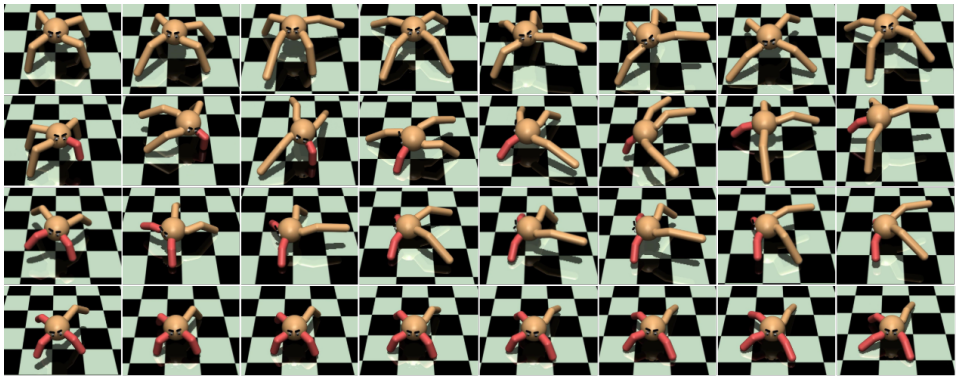

Figure 7: From top to bottom: healthy Ant, Ant with one broken leg, Ant with broken two legs, and Ant with broken three legs.

4.2 Transferability evaluation of reward function

4.2.1 Pre-experiment 2. Before verifying the transferability of the reward function generated by ST-IRL, we first verify its feasibility through a pre-experiment.

The goal of this pre-experiment is to determine whether ST-IRL can learn instructive information about the task from the expert data or not. The reward function maps the task information in the environment. Therefore, we can indirectly guide the agent to achieve the task goal through the reward function. We wonder if the reward function R_a generated by ST-IRL can learn the task information. We judge whether R_a and the reward function R_e set in the four Mujoco environments have similar scores for the same random action or not. It can be seen from Fig. 6 that the two reward functions indeed give similar scores with a consistent trend, proving that ST-IRL can learn task-oriented information from expert data.

4.2.2 Experiment 2-1. This section answers the problem: how transferable is the ST-IRL reward function for the same task with different agents?

We customize a healthy Ant and construct Ants with one, two and three broken legs, respectively, to meet the constraints of the same task and different agents. The experiment process is as follows: we first use the expert data of a healthy Ant to train the decoupled reward function in ST-IRL, and next use the obtained reward function as the reward function of the self-defined broken-legged Ant. Then, with known environmental rewards, we use forward reinforcement learning to train the SAC policy. Fig. 7 shows part of the frame data captured by a healthy Ant and Ants with bad legs walking through the SAC policy after training. It can be seen from the figure that Ants with one broken leg and two broken legs can still be used under the reward function of ST-IRL. However, Ant, with three broken legs, can no longer walk with only one leg. Fig. 8 shows the cumulative reward sums for the three broken-legged Ants during the training process. We can see that the more legs used in the custom Ant agent, the more valid information is learned

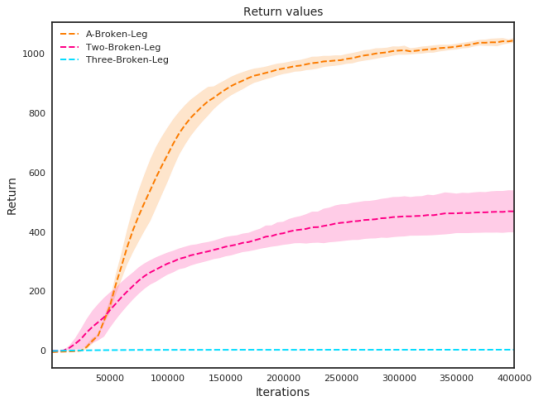

Figure 8: The cumulative rewards of three kinds of broken-legged Ants during training.

from the reward function. The result is that the further the distance traveled per unit of time, the higher the cumulative reward sum obtained.

4.2.3 Experiment 2-2. This section answers whether or not the reward function of ST-IRL can still be transferable in the same task in a different environment.

At first, we customize a PointMaze experimental environment. The goal of PointMaze is that the white ball should reach the top of the yellow ball through the shortest path. To demonstrate generality, we randomly select a scene from six different PointMaze environments as the original scene. And then, to fully demonstrate the stability and generalization ability of ST-IRL, we select relatively poor expert data in the original scenario. In the next step, we migrate the reward function learned by ST-IRL from the expert data to the following five variants of PointMaze, based on which we use the sac model for reinforcement learning training. Fig. 9 shows that the learned policy completes the task, demonstrating that the reward function generated by ST-IRL can generalize well to the same task in different environments.

5 CONCLUSION

Imitation learning and mainstream IRL suffer from poor results in the situation of complex environments and insufficient expert data. We propose the ST-IRL algorithm based on the Wasserstein metric and adversarial training to address this problem. Through experiments, we demonstrate the stability of the ST-IRL algorithm on policy and the transferability of the reward function. Furthermore, the proposed algorithm provides an idea for shaping the reward function in a complicated environments. Before constructing the reward function artificially, web collect expert data in that environment or similar environments, use the expert data to construct the base reward function in ST-IRL, and finally make appropriate fine-tuning to obtain the final reward function. This data-driven approach for constructing the reward function can avoid the subjectivity and limitations associated with artificially shaped rewards.

In the future, we will consider introducing the idea of hierarchical reward functions [31–33] into ST-IRL to improve the exploratory power and generalizability of the reward functions. Moreover, ST-IRL can be combined with meta-learning [34, 35] to use the reward functions generated by ST-IRL as metadata for further optimization.

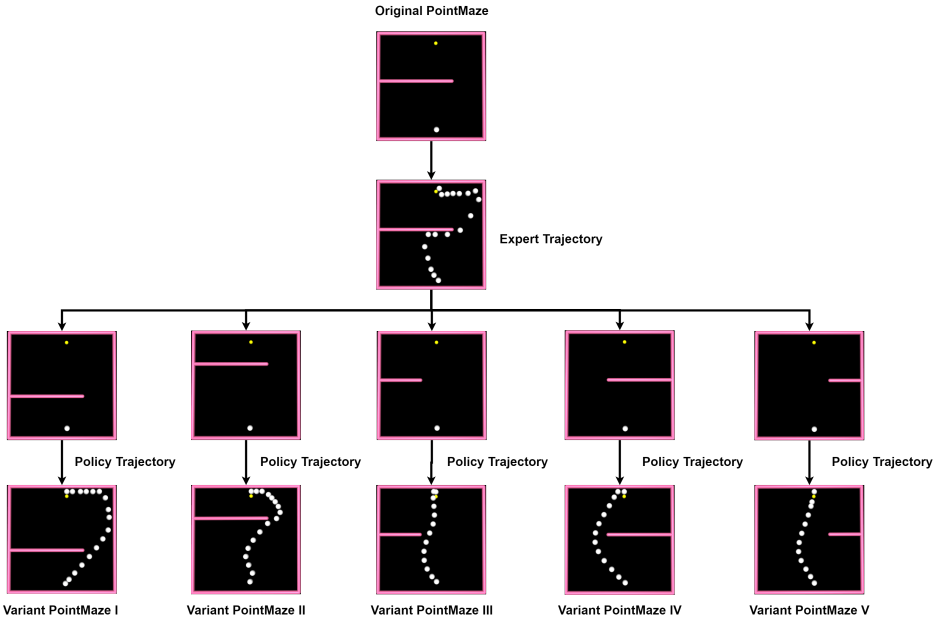

Figure 9: From top to bottom are the original PointMaze environment, the expert trajectory, the six variants of the PointMaze environment and the policy trajectories in the corresponding environments.

ACKNOWLEDGMENTS

This work was supported by CAS Project for Young Scientists in Basic Research, Grant No.YSBR-040.

REFERENCES

[1] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, GeorgeVan Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, *et al.* 2016. Mastering the game of go with deep neural networks and tree search. nature, 529(7587):484–489

[2] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki,Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. 2019. Dota 2 with large scale deep reinforcement learning. CoRR, abs/1912.06680

[3] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, *et al.* 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature, 575(7782):350–354

[4] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz,Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, *et al.* 2020. Learning dexterous in-hand manipulation. The InternationalJournal of Robotics Research, 39(1):3–20

[5] V Nguyen, SB Orbell, Dominic T Lennon, Hyungil Moon, Florian Vigneau, Leon C Camenzind, Liuqi Yu, Dominik M Zumbühl, G Andrew D Briggs, Michael A Osborne, *et al.* 2021. Deep reinforcement learning for efficient measurement of quantum devices. npj Quantum Information, 7(1):1–9

[6] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, *et al.* 2022. Magnetic control of tokamak plasmas through deep reinforcement learning. Nature, 602(7897):414–419

[7] Yin Yuxia Shuai Xiaoying and Zhang Bin. 2021. A fast convergencealohabased on reinforcement learning. International Journal of Computer Theoryand Engineering, 13(3):96–99

[8] Richard S Sutton and Andrew G Barto. 2018. Reinforcement learning: An introduction. MIT press, London,England

[9] Zhong Deng and Jane W.-S. Liu. 1997. Scheduling real-time applications in an open environment. In Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS'97), December 3-5, 1997, San Francisco, CA, USA, pages 308–319. IEEE ComputerSociety

[10] Christopher G Atkeson and Stefan Schaal. 1997. Robot learning from demonstration. In ICML, volume 97, pages 12–20

[11] Andrew Y Ng, Stuart Russell, *et al.* 2000. Algorithms for inverse reinforcement learning. In Icml, volume 1, page 2

[12] Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the twenty-first international conference on Machine learning, page 1

[13] Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 661–668. JMLR Workshop and Conference Proceedings

[14] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, *et al.* 2008. Maximum entropy inverse reinforcement learning. In Aaai, volume 8, pages 1433–1438. Chicago, IL, USA

[15] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. 2015. Deep inverse reinforcement learning. CoRR, abs/1906.05274

[16] Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. Guided cost learning: Deep inverse optimal control via policy optimization. In International conference on machine learning, pages 49–58. PMLR

[17] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. Advances in neural information processing systems, 29

[18] Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. 2020. A divergence minimization perspective on imitation learning methods. In Conference on Robot Learning, pages 1259–1277. PMLR

[19] Justin Fu, Katie Luo, and Sergey Levine. 2017. Learning robust rewards with adversarial inverse reinforcement learning. CoRR, abs/1710.11248

[20] Syed Mumtaz Ali and Samuel D Silvey. 1966. A general class of coefficients of divergence of one distribution from another. Journal of the Royal Statistical Society: Series B (Methodological), 28(1):131–142

[21] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, *et al.* 2018. Soft actor-critic algorithms and applications. CoRR, abs/1812.05905

[22] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In International conference on machine learning, pages 214–223. PMLR

[23] Abdeslam Boularias, Jens Kober, and Jan Peters. 2011. Relative entropy inverse reinforcement learning. In Proceedings of the fourteenth international conference on artificial intelligence and statistics, pages 182–189. JMLR Workshop and Conference Proceedings

[24] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. 2019. Efficient exploration via state marginal matching. CoRR, abs/1906.05274

[25] Minghuan Liu, Tairan He, Minkai Xu, and Weinan Zhang. 2021. Energy-based imitation learning. pages 809–817

[26] Tianwei Ni, Harshit Sikchi, Yufei Wang, Tejus Gupta, Lisa Lee, and Ben Eysenbach. 2021. f-irl: Inverse reinforcement learning via state marginal matching. In Conference on Robot Learning, pages 529–551. PMLR

[27] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. 2016. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. CoRR, abs/1611.03852

[28] Martin Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. CoRR, abs/1701.04862

[29] Zhiming Zhou, Jiadong Liang, Yuxuan Song, Lantao Yu, Hongwei Wang, Weinan Zhang, Yong Yu, and Zhihua Zhang. 2019. Lipschitz generative adversarial nets. In International Conference on Machine Learning, pages 7584–7593. PMLR

[30] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. Advances in neural information processing systems, 30

[31] Jonathan Sorg, Richard L Lewis, and Satinder Singh. 2010. Reward design via online gradient ascent. Advances in Neural Information Processing Systems, 23

[32] Xiaoxiao Guo, Satinder Singh, Richard Lewis, and Honglak Lee. 2016. Deep learning for reward design to improve monte carlo tree search in atari games. CoRR, abs/1604.07095

[33] Zeyu Zheng, Junhyuk Oh, and Satinder Singh. 2018. On learning intrinsic rewards for policy gradient methods. Advances in Neural Information Processing Systems, 31

[34] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In International conference on machine learning, pages 1126–1135. PMLR

[35] Ricardo Vilalta and Youssef Drissi. 2002. A perspective view and survey of meta-learning. Artificial intelligence review, 18(2):77–95

A A DERIVATION AND PROOF

Let w and w' be two parameter vectors in $\mathbb{R}^d$. Let P_e be a fixed distribution over $\mathbb{F}$. Let s be a random variable over another space $\mathbb{S}$. Let $f: \mathbb{S} \times \mathbb{R}^d \to \mathbb{F}$ be a function, that will be denoted $f_w(s)$ with the s the first coordinate and w the second. Let P_w denote the distribution of $f_w(\mathbb{S})$.

From the definition of the Wassertein metric, we know γ is the distribution of the joint $(f_w(\mathbb{S}),\ f_{w'}(\mathbb{S}))$, which clearly has $(P_w,\ P_{w'})$.

By the definition of the Wassertein metric, we have

$$\begin{aligned} W(P_w, P_{w'}) &= \inf_{\gamma \sim \Pi(\mathrm{P_w}, \mathrm{P_{w'}})} \mathbb{E}_{(x,y)\sim\gamma}[x-y] \\ &= \inf_{\gamma \sim \Pi(\mathrm{P_w}, \mathrm{P_{w'}})} \mathbb{E}_{(x,y)\sim\gamma}[y-x] \\ &\le \mathbb{E}_{(x,y)\sim\gamma}[x-y] \\ &= \mathbb{E}_S[f_w(s) - f_{w'}(s)] \end{aligned}$$

$= \mathbb{E}_S[f_{w'}(s) - f_w(s)]$ **(satisfy symmetry)**if f is continuous in w, then $(w \to w') \to (f_w \to f_{w'})$. So $f_w(s) - f_{w'}(s)$ pointwise as function of s.

Since $\mathbb{F}$ is compact, the distance of any two elements in it has to be uniformly bounded by some constant D, and therefore $f_w(s) - f_{w'}(s) \le D$ for all w and s uniformly. By the bounded convergence theorem, we therefore have

$$W(P_w, P_{w'}) \le= \mathbb{E}_S[f_{w'}(s) - f_w(s)] \to 0_{w \to w'}$$

So, we have

$$W(P_w, P_{w'}) = 0 \Leftrightarrow W(f_w(s), f_{w'}(s)) = 0$$

$$\Leftrightarrow f_w(s) = f_{w'}(s) \textbf{ (satisfy the identity of indiscernibles)}$$

Finally, we have that

$|W(\mathbb{P}_e, \mathbb{P}_w) - W(\mathbb{P}_e, \mathbb{P}_{w'})| \leq W(\mathbb{P}_w, \mathbb{P}_{w'})$ **(satisfy triangle inequality)**

Next, we will prove the continuity and differentiable of Wassertein metric.

Let f be locally Lipschitz. Then, for a given pair (w, s) there is a constant $L(w, s)$ and an open set U such that $(w, s) \in U$, such that for every $(w', s') \in U$ we have

$$f_w(s) - f_{w'}(s') \leq L(w, s)\left(w - w' + s - s'\right)$$

because $(w', s) \in U$, so when take expections, we have $s' = s$ and

$$\mathbb{E}_S\left[f_w(s) - f_{w'}(s)\right] \leq w - w'\mathbb{E}_S\left[L(w, s)\right]$$

Therefore, we can define $U_w = \{w' | w', s) \in U\}$. It is easy to see that since U is open, U_w is as well. Furthermore, we can define $L(w) = \mathbb{E}_S[L(w, s)$ and achieve

$$|W(\mathbb{P}_e, \mathbb{P}_w) - W(\mathbb{P}_e, \mathbb{P}_{w'})| \leq W(\mathbb{P}_w, \mathbb{P}_{w'}) \leq L(w)\, w - w'$$

for all $w' \in U_w$ meaning that $W(\mathbb{P}_e, \mathbb{P}_w)$ is locally Lipschitz. This obviously implies that $W(\mathbb{P}_e, \mathbb{P}_w)$ is **everywhere continuous**, and by Radamacher's theorem we know it has to be **differentiable almost everywhere.**