CWAE-IRL: Formulating a supervised approach to Inverse Reinforcement Learning problem

Anonymous authors
Paper under double-blind review

Abstract

Inverse reinforcement learning (IRL) is used to infer the reward function from the actions of an expert running a Markov Decision Process (MDP). A novel approach using variational inference for learning the reward function is proposed in this research. Using this technique, the intractable posterior distribution of the continuous latent variable (the reward function in this case) is analytically approximated to appear to be as close to the prior belief while trying to reconstruct the future state conditioned on the current state and action. The reward function is derived using a well-known deep generative model known as Conditional Variational Auto-encoder (CVAE) with Wasserstein loss function, thus referred to as Conditional Wasserstein Auto-encoder-IRL (CWAE-IRL), which can be analyzed as a combination of the backward and forward inference. This can then form an efficient alternative to the previous approaches to IRL while having no knowledge of the system dynamics of the agent. Experimental results on standard benchmarks such as objectworld and pendulum show that the proposed algorithm can effectively learn the latent reward function in complex, high-dimensional environments.

1 Introduction

Reinforcement learning, formalized as Markov decision process (MDP), provides a general solution to sequential decision making, where given a state, the agent takes an optimal action by maximizing the long-term reward from the environment Bellman (1957). However, in practice, defining a reward function that weighs the features of the state correctly can be challenging, and techniques like reward shaping are often used to solve complex real-world problems Ng et al. (1999). The process of inferring the reward function given the demonstrations by an expert is defined as inverse reinforcement learning (IRL) or apprenticeship learning Ng et al. (2000); Abbeel & Ng (2004).

The fundamental problem with IRL lies in the fact that the algorithm is under defined and infinitely different reward functions can yield the same policy Finn et al. (2016). Previous approaches have used preferences on the reward function to address the non-uniqueness. Ng et al. (2000) suggested reward function that maximizes the difference in the values of the expert's policy and the second best policy. Ziebart et al. (2008) adopted the principle of maximum entropy for learning the policy whose feature expectations are constrained to match those of the expert's. Ratliff et al. (2006) applied the structured max-margin optimization to IRL and proposed a method for finding the reward function that maximizes the margin between expert's policy and all other policies. Neu & Szepesvári (2009) unified a direct method that minimizes deviation from the expert's behavior and an indirect method that finds an optimal policy from the learned reward function using IRL. Syed & Schapire (2008) used a game-theoretic framework to find a policy that improves with respect to an expert's.

Another challenge for IRL is that some variant of the forward reinforcement learning problem needs to be solved in a tightly coupled manner to obtain the corresponding policy, and then compare this policy to the demonstrated actions Finn et al. (2016). Most early IRL algorithms proposed solving an MDP in the inner loop Ng et al. (2000); Abbeel & Ng (2004); Ziebart et al. (2008). This requires perfect knowledge of the expert's dynamics which are almost always impossible to have. Several works have proposed to relax this requirement, for example by learning a value function instead of

a cost Todorov (2007), solving an approximate local control problem Levine & Koltun (2012) or generating a discrete graph of states Byravan et al. (2015). However, all these methods still require some partial knowledge of the system dynamics.

Most of the early research in this field has expressed the reward function as a weighted linear combination of hand selected features Ng et al. (2000); Ramachandran & Amir (2007); Ziebart et al. (2008). Non-parametric methods such as Gaussian Processes (GPs) have also been used for potentially complex, nonlinear reward functions Levine et al. (2011). While in principle this helps extend the IRL paradigm to flexibly account for non-linear reward approximation; the use of kernels simultaneously leads to higher sample size requirements. Universal function approximators such as non-linear deep neural network have been proposed recently Wulfmeier et al. (2015); Finn et al. (2016). This moves away from using hand-crafted features and helps in learning highly non-linear reward functions but they still need the agent in the loop to generate new samples to "guide" the cost to the optimal reward function. Fu et al. (2017) has recently proposed deriving an adversarial reward learning formulation which disentangles the reward learning process by a discriminator trained via binary regression data and uses policy gradient algorithms to learn the policy as well.

The Bayesian IRL (BIRL) algorithm proposed by Ramachandran & Amir (2007) uses the expert's actions as evidence to update the prior on reward functions. The reward learning and apprenticeship learning steps are solved by performing the inference using a modified Markov Chain Monte Carlo (MCMC) algorithm. Zheng et al. (2014) described an expectation-maximization (EM) approach for solving the BIRL problem, referring to it as the Robust BIRL (RBIRL). Variational Inference (VI) has been used as an efficient and alternative strategy to MCMC sampling for approximating posterior densities Jordan et al. (1999); Wainwright et al. (2008). Variational Auto-encoder (VAE) was proposed by Kingma & Welling (2014) as a neural network version of the approximate inference model. The loss function of the VAE is given in such a way that it automatically tries to maximize the likelihood of the data given the current latent variables (reconstruction loss), while encouraging the latent variables to be close to our prior belief of how the variables should look like (Kullbeck-Liebler divergence loss). This can be seen as an generalization of EM from maximum a-posteriori (MAP) estimation of the single parameter to an approximation of complete posterior distribution. Conditional VAE (CVAE) has been proposed by Sohn et al. (2015) to develop a deep conditional generative model for structured output prediction using Gaussian latent variables. Wasserstein Auto-Encoder (WAE) has been proposed by Tolstikhin et al. (2017) to utilize Wasserstein loss function in place of KL divergence loss for robustly estimating the loss in case of small samples, where VAE fails.

This research is motivated by the observation that IRL can be formulated as a supervised learning problem with latent variable modelling. This intuition is not unique. It has been proposed by Klein et al. (2013) using the Cascaded Supervised IRL (CSI) approach. However, CSI uses non-generalizable heuristics to classify the dataset and find the decision rule to estimate the reward function. Here, I propose to utilize the CVAE framework with Wasserstein loss function to determine the non-linear, continuous reward function utilizing the expert trajectories without the need for system dynamics. The encoder step of the CVAE is used to learn the original reward function from the next state conditioned on the current state and action. The decoder step is used to recover the next state given the current state, action and the latent reward function.

The likelihood loss, composed of the reconstruction error and the Wasserstein loss, is then fed to optimize the CVAE network. The Gaussian distribution is used here as the prior distribution; however, Ramachandran & Amir (2007) has described various other prior distributions which can be used based on the class of problem being solved. Since, the states chosen are supplied by the expert's trajectories, the CWAE-IRL algorithm is run only on those states without the need to run an MDP or have the agent in the loop. Two novel contributions are made in this paper:

- Proposing a generative model such as an auto-encoder for estimating the reward function leads to a more effective and efficient algorithm with locally optimal, analytically approximate solution.

- Using only the expert's state-action trajectories provides a robust generative solution without any knowledge of system dynamics.

Section 2 gives the background on the concepts used to build our model; Section 3 describes the proposed methodology; Section 4 gives the results and Section 5 provides the discussion and conclusions.

2 PRELIMINARIES

2.1 MARKOV DECISION PROCESS (MDPS)

In the reinforcement learning problem, at time t, the agent observes a state, $s_t \in S$, and takes an action, $a_t \in A$; thereby receiving an immediate scalar reward r_t and moving to a new state s_{t+1}. The model's dynamics are characterized by state transition probabilities $p(s_{t+1}|s_t, a_t)$. This can be formally stated as a Markov Decision Process (MDP) where the next state can be completely defined by the previous state and action (Markov property) and the agent receives a scalar reward for executing the action Bellman (1957).

The goal of the agent is to maximize the cumulative reward (discounted sum of rewards) or value function:

$$v_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{1}$$

where $0 \le \gamma \le 1$ is the discount factor and r_t is the reward at time-step t.

In terms of a policy $\pi : S \to A$, the value function can be given by Bellman equation as:

$$v_\pi(s_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | S = s_t \right] \tag{2}$$

$$= \mathbb{E}_\pi \left[r_t + \sum_{k=1}^{\infty} \gamma^k r_{t+k} | S = s_t \right] \tag{3}$$

$$= \sum_a \pi(a_t|s_t) \sum_{s_{t+1}} p(s_{t+1}|s_t, a_t)$$

$$\left[r(s_t, a_t, s_{t+1}) + \gamma \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | S = s_{t+1} \right] \right] \tag{4}$$

$$= \sum_a \pi(a_t|s_t) \sum_{s_{t+1}} p(s_{t+1}|s_t, a_t)$$

$$\left[r(s_t, a_t, s_{t+1}) + \gamma v_\pi(s_t + 1) \right] \tag{5}$$

Using Bellman's optimality equation, we can define, for any MDP, a policy π is greater than or equal to any other policy π' if value function $v_\pi(s_t) \le v_{\pi'}(s_t)$ for all $s_t \in S$. This policy is known as an optimal policy (π^*) and its value function is known as optimal value function (v^*).

2.2 BAYESIAN IRL

The bayesian approach to IRL was proposed by Ramachandran & Amir (2007) by encoding the reward function preference as a prior and optimal confidence of the behavior data as the likelihood. Considering the expert χ is executing an MDP $M = (S, A, p, \gamma)$, the reward for χ is assumed to be sampled from a prior (known) distribution P_R defined as:

$$P_R(R) = \prod_{s \in S, a \in A} P(R(s, a)) \tag{6}$$

The distribution to be used as a prior depends on the type of problem. The expert's goal of maximizing accumulated reward is equivalent to finding the optimal action of each state. The likelihood thus

defines our confidence in χ's ability to select the optimal action. This is modeled as a exponential distribution for the likelihood of trajectory τ with Q^* as:

$$Pr_\chi(\tau|R) = \frac{1}{Z'} \exp^{\alpha_\chi Q^*(\tau,R)} \tag{7}$$

where α_χ is a parameter representing the degree of confidence in χ's ability. The posterior probability of the reward function R is computed using Bayes theorem,

$$Pr_\chi(R|\tau) = \frac{Pr_\chi(\tau|R)P_R(R)}{Pr(\tau)} \tag{8}$$

$$= \frac{1}{Z} \exp^{\alpha_\chi Q^*(\tau,R)} P_R(R) \tag{9}$$

BIRL uses MCMC sampling to compute the posterior mean of the reward function.

2.3 VARIATIONAL INFERENCE

For observations $x = x_{1:n}$ and latent variables $z = z_{1:m}$, the joint density can be written as:

$$p(z,x) = p(z)p(x|z) \tag{10}$$

The latent variables are drawn from a prior distribution $p(z)$ and they are then related to the observations through the likelihood $p(x|z)$. Inference in a bayesian framework amounts to conditioning on data and computing the posterior $p(z|x)$. In lot of cases, this posterior is intractable and requires approximate inference. Variational inference has been proposed in the recent years as an alternative to MCMC sampling by using optimization instead of sampling Blei et al. (2017).

For a family of approximate densities ζ over the latent variables, we try to find a member of the family that minimizes the Kullback-Leibler (KL) divergence to the exact posterior

$$q^*(z) = \underset{q(z)\in\zeta}{argmin} KL(q(z)||p(z|x)) \tag{11}$$

The posterior is then approximated with the optimized member of the family $q^*(z)$. The KL divergence is then given by

$$KL(q(z)||p(z|x)) = \mathbb{E}[log(q(z))] - \mathbb{E}[log(p(z|x))] \tag{12}$$

$$= \mathbb{E}[log(q(z))] - \mathbb{E}[log(p(z,x))] + log(p(x)) \tag{13}$$

Since, the divergence cannot be computed, an alternate objective is optimized in VAE called evidence lower bound (ELBO) that is equivalent,

$$L_{ELBO}(q) = \mathbb{E}[log(p(x))] - KL(q(z)||p(z|x) \tag{14}$$

$$= \mathbb{E}[log(p(x|z))] - KL(q(z|x)||p(z|x) \tag{15}$$

This can be defined as a sum of two separate losses:

$$L_{VAE} = L_{lk} + L_{div} \tag{16}$$

where L_{lk} is the loss related to the log-likelihood and L_{div} is the loss related to the divergence.

CVAE is used to perform probabilistic inference and predict diversely for structured outputs. The loss function is slightly altered with the introduction of class labels c:

$$L_{CVAE} = \mathbb{E}[log(p(x|c)] - KL(q(z|x,c)||p(z|x,c)) \tag{17}$$

$$= \mathbb{E}[log(p(x|z,c))] - KL(q(z|x,c)||p(z|x,c)) \tag{18}$$

2.4 WASSERSTEIN LOSS FUNCTION

Wasserstein distance, also known as Kantorovich-Rubenstein distance or earth mover's distance (EMD) Rubner et al. (2000), provides a natural distance over probability distributions in the metric space Frogner et al. (2015). It is a formulation of optimal transport problem Villani (2008) where the Wasserstein distance is the minimum cost required to move a pile of earth (an arbitrary distribution) to another. The mathematical formulation given by Kantorovich Tolstikhin et al. (2017) is:

$$W_c(P_X, P_Y) = \underset{\Gamma \in P(X \sim P_X, Y \sim P_Y)}{inf} \mathbb{E}_{(X,Y) \sim \Gamma}[c(X,Y)] \tag{19}$$

where $c(X,Y)$ is the cost function, X and Y are random variables with marginal distributions P_X and P_Y respectively.

EMD has been utilized in various practical applications in computer science such as pattern recognition in images He et al. (2018). Wasserstein GAN (WGAN) has been proposed by Arjovsky et al. (2017) to minimize the EMD between the generative distribution and the data distribution. Tolstikhin et al. (2017) proposed Wasserstein Auto-encoder (WAE) where the divergence loss has been calculated using the EMD instead of KL-divergence and has been shown to be robust in presence of noise and smaller samples.

3 Conditional Wasserstein Auto-Encoder-IRL

In this paper, my primary argument is that the inverse reinforcement learning problem can be devised as a supervised learning problem with learning of latent variable. The reward function, $r(s_t, a_t, s_{t+1})$, can be formulated as a latent function which is dependent on the state at time t, s_t, action at time t, a_t, and state at time $(t+1)$, s_{t+1}. In the CVAE framework, using the state and action pair as the class label c and rewriting the CVAE loss in Equation 17 with s_{t+1} as x and reward at time t, r_t as z, we get:

$$L_{CVAE-IRL} = \mathbb{E}[log(p(s_{t+1}|s_t, a_t))] - KL(q(r_t|s_{t+1}, s_t, a_t)||p(r_t|s_{t+1}, s_t, a_t)) \qquad (20)$$

The first part of Equation 20 provides the log likelihood of transition probability of an MDP and the second part gives the KL-divergence of the encoded reward function to the prior gaussian belief. Thus, the proposed method tries to recover the next state from the current state and current action by encoding the reward function as the latent variable and constraining the reward function to lie as close to the gaussian function. The network structure of the method is given in Figure 1.
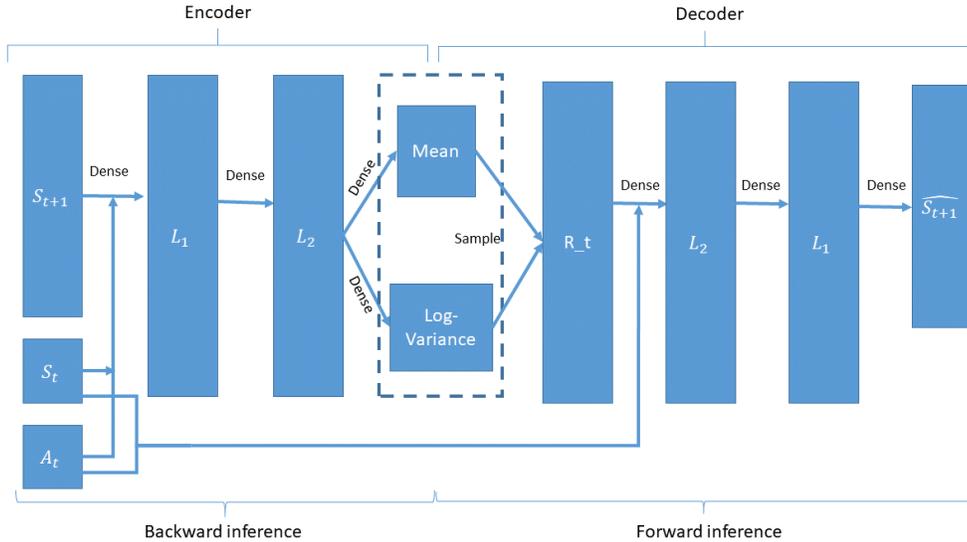

Figure 1: CWAE-IRL Network architecture

3.1 Encoder

The encoder is a neural network which inputs the current state, current action and next state and generates a probability distribution $q(r_t|s_{t+1}, s_t, a_t)$, assumed to be isotropic gaussian. Since a near-optimal policy is inputted into the IRL framework, minibatches of randomly sampled (s_{t+1}, s_t, a_t) are introduced into the network. Two hidden layers with dropout are used to encode the input data into a latent space, giving two outputs corresponding to the mean and log-variance of the distribution. This step is similar to the backward inference problem in IRL methodology where the reward function is constructed from the sampled trajectories for a near-optimal agent.

3.2 DECODER

Given the current state and action, the decoder maps the latent space into the state space and reconstructs the next state $\hat{s}_{t+1}$ from a sampled r_t (from the normal distribution). Similar to the VAE formulation, samples are generated from a standard normal distribution and reparameterized using the mean and log-variance computed in the encoder step. This step resembles the forward inference problem of an MDP where given a state, action and reward distribution, we estimate the next state that the agent gets to. Two hidden layers with dropout are used similar to the encoder.

3.3 USING WASSERSTEIN LOSS

Even though the KL-divergence should be able to provide for the loss theoretically, it does not converge in practice and indeed gives really large values in case of small samples such as in our formulation. Tolstikhin et al. (2017) provides a Maximum Mean Discrepancy (MMD) measure based on Wasserstein metric for a positive-definite reproducing kernel $k(\cdot, \cdot)$ such as the Radial Basis Function (RBF) kernel:

$$MMD_k(P_z, Q_z) = \left\| \int_z k(z, \cdot)dP_z(z) - \int_z k(z, \cdot)dQ_z(z) \right\|_{H_k} \tag{21}$$

where H_k is the Reproducing Kernel Hilbert Space (RKHS) mapping $z : \mathbb{Z} \to \mathbb{R}$. The divergence loss can then be written as:

$$W(q(z|x)||p(z|x)) = \frac{\lambda}{n(n-1)} \sum_{l \neq j} k(z_l, z_j) + \frac{\lambda}{n(n-1)} \sum_{l \neq j} k(\tilde{z}_l, \tilde{z}_j)$$
$$- \frac{2\lambda}{n^2} \sum_{l,j} k(z_l, \tilde{z}_j) \tag{22}$$

where c is the cost between the input, x and the output, $\tilde{z}$, of the decoder, D, using the sampled latent variable, z (given as mean squared error), The resulting CWAE-IRL loss function is given as:

$$L_{CWAE-IRL} = \mathbb{E}[log(p(s_{t+1}|s_t, a_t))] - W(q(r_t|s_{t+1}, s_t, a_t)||p(r_t|s_{t+1}, s_t, a_t)) \tag{23}$$

4 EXPERIMENTS

In this section, I present the results of CWAE-IRL on two simulated tasks, objectworld and pendulum.

4.1 OBJECTWORLD

Objectworld is a generalization of gridworld Sutton & Barto (1998), described in Levine et al. (2011). It contains NxN grid of states with five actions per state, corresponding to steps in each direction and staying in place. Each action has a 30% chance of moving in a different random direction. There are randomly assigned objects, having one of 2 inner and outer colors chosen, red and green. There are 4 continuous features for each of the grids, each giving the Euclidean distance to the nearest object with a specific inner or outer color. The true reward is +1 in states within 3 cells of outer red and 2 cells of outer green, -1 within 3 cells of outer red, and zero otherwise. Inner colors serve as distractors. The expert trajectories fed have a sample length of 16. The algorithms for objectworld, Maximum Entropy IRL and Deep Maximum Entropy IRL are used from the GitHub implementation of Alger (2016) without any modifications. Only continuous features are used for all implementations.

CWAE-IRL is compared with prior IRL methods, described in the previous sections. Among prior methods chosen, CWAE-IRL is compared to BIRL Ramachandran & Amir (2007), Maximum Entropy IRL Ziebart et al. (2008) and Deep Maximum Entropy IRL Wulfmeier et al. (2015). Only the Maximum Entropy IRL uses the reward as a linear combination of features while the others describe it in a non-linear fashion. The learnt rewards for all the algorithms are shown in Figure 2 with an objectworld of grid size 10. CWAE-IRL can recover the original reward distribution while the Deep Maximum Entropy IRL overestimates the reward in various places. Maximum Entropy IRL and
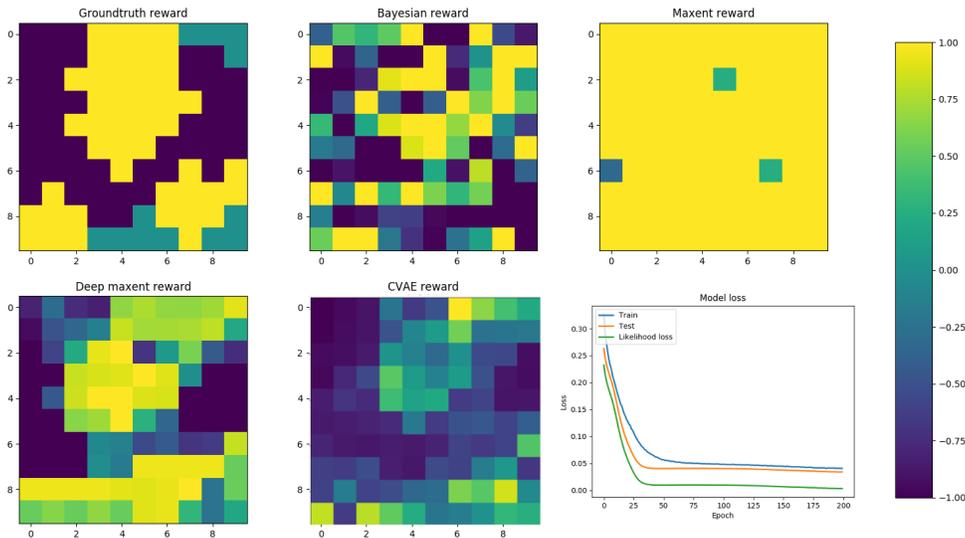
Figure 2: Reward comparison of IRL algorithms for an objectworld of grid size 10 with random placement of objects. (a) Groundtruth reward (b) Bayesian IRL reward (c) Maximum entropy reward (d) Deep maximum entropy reward (e) VAE reward (f) Training loss (blue), validation loss (red) and likelihood loss (green) over epochs

BIRL completely fail to learn the rewards. Deep Maximum Entropy tends to give negative rewards to state spaces which are not well traversed in the example trajectories. However, CWAE-IRL generalizes over the spaces even though they have not been visited frequently. Also, due to the constraint of being close to the prior gaussian belief, the scale of rewards are best captured by the proposed algorithm as compared to the other algorithms which tend to overscale the rewards.

4.2 PENDULUM

The pendulum environment Brockman et al. (2016) is an well-known problem in the control literature in which a pendulum starts from a random position and the goal is to keep it upright while applying the minimum amount of force. The state vector is composed of the cosine (and sine) of the angle of the pendulum, and the derivative of the angle. The action is the joint effort as 11 discrete actions linearly spaced within the $[-2, 2]$ range. The reward is

$$R = -(w_1 \cdot \|\theta\|^2 + w_2 \cdot \left\|\dot{\theta}\right\|^2 + w_3 \cdot \|a\|^2), \tag{24}$$

where w_1, w_2 and w_3 are the reward weights for the angle θ, derivative of angle $\dot{\theta}$ and action a respectively. The optimal reward weights given by OpenAI are $[1, 0.1, 0.001]$ respectively. An episode is limited to 1000 timesteps.

A deep Q-network (DQN) has been proposed by Mnih et al. (2015) that combines deep neural networks with RL to solve continuous state discrete action problems. DQN uses a neural network with gives the Q-values for every action and uses a buffer to store old states and actions to sample from to help stabilize training. Using a continuous state space makes it impossible to have all states visited during training. This also makes it very difficult for the comparison of recovered reward with the actual reward. The DQN is trained for 50,000 episodes. The CWAE-IRL is trained using 25 trajectories and the reward is predicted for 5 trajectories. The error plot between the reward recovered and the actual reward is given in Figure 3. The mean error hovers around 0 showing that under for majority of the states and actions, the proposed method is able to recover the correct reward.
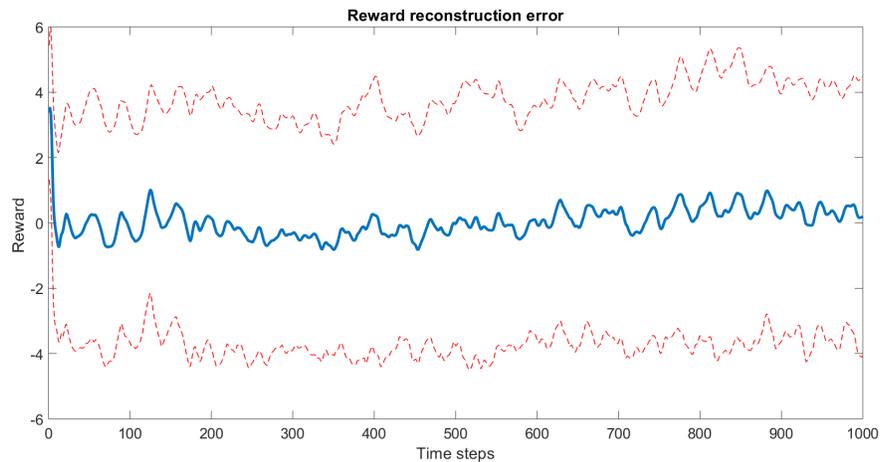
Figure 3: Reward error for predicted minus the actual error recovered at each time step (smoothed) with 1 σ confidence interval for the pendulum environment

5 CONCLUSIONS

I have presented an algorithm for inverse reinforcement learning which learns the latent reward function using a conditional variational auto-encoder with Wasserstein loss function. It learns the reward function as a continuous, Gaussian distribution while trying to reconstruct the next state given the current state and action. The proposed model makes the inference process scalable while making it easier for inference of reward functions. Inferring a continuous parametric distribution of the reward functions can prove useful for classifying behaviors of decision making agents in diverse applications such as autonomous driving.

REFERENCES

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1. ACM, 2004.

Matthew Alger. Inverse reinforcement learning, 2016. URL https://doi.org/10.5281/zenodo.555999.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, pp. 679–684, 1957.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Arunkumar Byravan, Mathew Monfort, Brian D Ziebart, Byron Boots, and Dieter Fox. Graph-based inverse optimal control for robot manipulation. In *Ijcai*, volume 15, pp. 1874–1890, 2015.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pp. 49–58, 2016.

Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya, and Tomaso A Poggio. Learning with a wasserstein loss. In *Advances in Neural Information Processing Systems*, pp. 2053–2061, 2015.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Ran He, Xiang Wu, Zhenan Sun, and Tieniu Tan. Wasserstein cnn: Learning invariant features for nir-vis face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 41(7): 1761–1773, 2018.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, 2014.

Edouard Klein, Bilal Piot, Matthieu Geist, and Olivier Pietquin. A cascaded supervised learning approach to inverse reinforcement learning. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 1–16. Springer, 2013.

Sergey Levine and Vladlen Koltun. Continuous inverse optimal control with locally optimal examples. *arXiv preprint arXiv:1206.4617*, 2012.

Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 19–27, 2011.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Gergely Neu and Csaba Szepesvári. Training parsers by inverse reinforcement learning. *Machine learning*, 77(2-3):303, 2009.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pp. 663–670, 2000.

Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. *Urbana*, 51 (61801):1–4, 2007.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 729–736. ACM, 2006.

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pp. 3483–3491, 2015.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Umar Syed and Robert E Schapire. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, pp. 1449–1456, 2008.

Emanuel Todorov. Linearly-solvable markov decision problems. In *Advances in neural information processing systems*, pp. 1369–1376, 2007.

Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Deep inverse reinforcement learning. *CoRR, abs/1507.04888*, 2015.

Jiangchuan Zheng, Siyuan Liu, and Lionel M Ni. Robust bayesian inverse reinforcement learning with sparse behavior noise. In *AAAI*, pp. 2198–2205, 2014.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.