Robust Maximum Entropy Behavior Cloning

Mostafa Hussein
Cognitive Assistive Robotics Lab
University of New Hampshire
Durham, NH 03801
mhussein@cs.unh.edu

Brendan Crowe
Department of Statistics
University of New Hampshire
Durham, NH 03801
bjc1041@wildcats.unh.edu

Marek Petrik
Department of Computer Science
University of New Hampshire
Durham, NH 03801
mpetrik@cs.unh.edu

Momotaz Begum
Cognitive Assistive Robotics Lab
University of New Hampshire
Durham, NH 03801
mbegum@cs.unh.edu

Abstract

Imitation learning (IL) algorithms use expert demonstrations to learn a specific task. Most of the existing approaches assume that all expert demonstrations are reliable and trustworthy, but what if there exist some adversarial demonstrations among the given data-set? This may result in poor decision-making performance. We propose a novel general frame-work to directly generate a policy from demonstrations that autonomously detect the adversarial demonstrations and exclude them from the data set. At the same time, it's sample, time-efficient, and does not require a simulator. To model such adversarial demonstration we propose a min-max problem that leverages the entropy of the model to assign weights for each demonstration. This allows us to learn the behavior using only the correct demonstrations or a mixture of correct demonstrations.

1 Introduction and Related Work

Imitation learning IL addresses the problem of learning a policy from demonstrations provided by an expert [5, 17]. As robots become more involved in our daily lives, the ability to program robots and teach them new skills becomes significantly more important. The ability of a robot to effectively learn from demonstrations would greatly increase the quality of robotics applications. A common assumption in most IL approaches is that all expert demonstrations are reliable and trustworthy, but that is not always the case. In this paper we address the problem of adversarial demonstration and how we can detect those demonstrations in any given data-set. Before we go further we want to define what an adversarial demonstration is and why it might exist in a data-set. It is any demonstration that does not follow the optimal policy/policies defined by the task expert.

There are two main approaches for IL: inverse reinforcement learning (IRL), where we learn a reward function that the demonstrator is trying to maximize during the task, then generating a policy that maximizes the generated reward [15, 21]. More recent approaches [7, 11], draw a connection between IRL and generative adversarial networks [6, 9] and managed to get better expected return than the classical IRL algorithms. The application of these new techniques in practice is often hindered by the need for millions of samples during training to converge even in the simplest control tasks [13].

The second approach is behavioral cloning (BC), the goal in BC is to learn a mapping between the states and actions as a supervised learning problem [18]. BC is considered conceptually simple and theoretically sound [26]. The main criticism for BC in its current state is the covariate shift [19, 20]. One of the main advantages of BC over IRL is, it does not require a simulator or extra samples during the learning. To be able to deploy a robot and safely use it in our daily lives, we must have the ability to teach the robot new tasks without the need for a simulator to sample from, as well as considering

NeurIPS 2020 3rd Robot Learning Workshop: Grounding Machine Learning Development in the Real World.

the time efficiency. This feature is only feasible using BC and that is our main reason behind building our approach upon BC.

A few works like [27] assume the existence of noisy demonstrations and propose a Bayesian approach to detect them, the authors use a latent variable to assign a weight to each data point in the demonstration set and find these weights using an EM-like algorithm. Criticism of this approach is that they use an assumption over a prior distribution which is mostly task dependent and they can only handle until 10% of the data is random noise, and cannot handle structured adversarial behavior. Other approaches in IRL like [10, 23] use the "failed" demonstration to train the model beside the correct ones, but they assume that these failed demonstrations are given and labeled in the demonstration set.

In this paper, we propose a novel robust probabilistic IL frame-work that has the ability to autonomously detect the adversarial demonstrations and exclude it from the training data-set. Robust Maximum ENTropy (RM-ENT), is a frame-work that defines the demonstrated task by constraining the feature expectation matching between the demonstration and the generated model. The feature matching constraint by itself cannot generate a policy and here is where the maximum entropy principles [2, 12] will play the main role in our frame-work. (1) It will choose the model among the task model space that has the maximum entropy; (2) Simultaneously it will analyze the entropy contributed by each demonstration and will set weights to each demonstration that distinguishes between the correct and adversarial ones. We demonstrate that RM-ENT achieves better expected return and robustness than existing IRL and standard BC in classical control tasks in the OpenAi-gym simulator [4].

2 Preliminaries and Base Model

We use a tuple $(\mathcal{S}, \mathcal{A}, \rho_0)$ to define an infinite horizon Markov process (MDP), where $\mathcal{S}$ represents the state space, $\mathcal{A}$ represents the action space, $\rho_0 : \mathcal{S} \to \mathbb{R}$ is the distribution of the initial state s_0. Let π denote a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ and π_E denote the expert policy we have from the demonstrations. The expert demonstrations $\mathcal{D}$ are a set of trajectories, each of which consists of a sequence of state-action pairs $\mathcal{D} = (a_i, s_i)_{i=1}^{Q}$ where Q is the number of state-action pairs in each demonstration.

In most IL algorithms we try to represent the task using a set of features $f_i(s, a), i \in \{1, 2, \ldots, n\}$ that contain enough information to help us solve the IL problem while limiting the complexity of learning. Now comes the most common questions in the IL problem: *What should we match between the expert and the learner?* Many answers have been introduced among the IL community but the most successful approach until now is the *feature expectation matching (FEM)* [1, 7, 24, 28]:

$$\begin{aligned} \mathbb{E}_{\tilde{\pi}}[f_i] &= \mathbb{E}_{\pi}[f_i], i \in \{1, 2, \ldots, n\} \\ \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \tilde{p}(s)\tilde{\pi}(a|s) f_i(s, a) &= \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \tilde{p}(s)\pi(a|s) f_i(s, a) \end{aligned} \tag{1}$$

Where $\tilde{p}$ is the state-action expert distribution while p is the learned model and $\tilde{p}(s)$ is the expert distribution of s in the demonstration set.

FEM by itself is an ill-defined problem that cannot generate a policy in the case of BC or a reward function in IRL, since there are many optimal policies that can explain a set of demonstrations, and many rewards that can explain an optimal policy.

We use the principles of maximum entropy [12] to solve the ambiguity among the model space where we are looking for the model that had the maximum entropy with the constraint of FEM.

$$\begin{aligned} \max_{\pi \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}} \quad & H(\pi) \equiv -\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \tilde{p}(s)\pi(a|s) \log \pi(a|s) \\ \text{s.t.} \quad & \mathbb{E}_{\tilde{\pi}}[f_i] - \mathbb{E}_{\pi}[f_i] = 0 \quad i = 1, \ldots, n \\ & \sum_{a \in \mathcal{A}} \pi(a|s) - 1 = 0 \quad \forall\, s \in \mathcal{S} \end{aligned} \tag{2}$$

Using a Lagrange multiplier we can solve this convex problem and get a generalized form for the policy.[1] Using the previous formulation we manage to generate a policy using only a few

[1] A complete derivation can be found in Appendix A.

demonstrations because it depends on the feature itself not on how many data points we have, which will be shown in the result section.

3 Robust Maximum Entropy Behavior Cloning (RM-ENT)

In the previous section, we introduced how to learn the best fit model from our set of demonstrations, but the assumption was that those demonstrations are coming from the expert without any noise or inaccurate trajectories which is not the case in real-life applications. Our goal here is to be able to use only the set of the demonstration that can lead us to the optimal policy and exclude anything else.

Now we will introduce how we can add robustness to our model. We will add the w variable which is a weight that is given to each demonstration. The goal is to give the adversarial demonstration the minimum possible weight and to give the correct demonstration a higher weight automatically through the learning. The main hypothesis is coming from maximum entropy principles. The original definition of entropy is the average level of uncertainty inherent in the random variable. So we can say that we are looking for the demonstrations that add the least amount of entropy to the model.

We can explain more by saying, if we have an adversarial demonstration it will try to add incorrect, or "random", information to the model which will increase its entropy. So the goal is to limit this adversarial demonstration by assigning a lower weight to it. At the same time, if two demonstrations add the same amount of information to the model, they should have the same weight. Based on the previous discussion we will introduce these two new notations:

$$\tilde{p}_w(s) = \frac{1}{M}\sum_{d=1}^{D} w_d \cdot \tilde{p}(s|d) \qquad (3a) \qquad\qquad \tilde{\pi}_w(s|a) = \frac{1}{M}\sum_{d=1}^{D} w_d \cdot \tilde{\pi}(s,a|d) \qquad (3b)$$

Where D is the total number of demonstrations, and M should be $\sum_{d=1}^{D} w_d$. Which is the minimum number of demonstrations that we can trust in the given set.

By modifying (5) with the new variable w we will get our primal problem as follows:

$$\begin{aligned}
\min_{w\in\mathbb{R}^D}\max_{\pi\in\mathbb{R}^{\mathcal{S}\times\mathcal{A}}} \quad & -\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \pi(a|s)\log\pi(a|s)\sum_{d=1}^{D} w_d\cdot\tilde{p}(s,d) \\
\text{s. t.} \quad & \sum_{d=1}^{D} w_d \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} f_i(s,a)\tilde{p}(s,d)\Big(\pi(a|s)-\tilde{\pi}(a|s,d)\Big)=0, \quad i=1,\dots,N \qquad [\pi] \\
& \sum_{a\in\mathcal{A}}\pi(a|s)-1=0, \quad \forall s\in\mathcal{S} \qquad [\pi] \\
& \sum_{d=1}^{D} w_d = M, \quad w_d\geq 0, \quad \forall d\in\mathcal{D}, \quad w_d\leq 1 \quad \forall d=1,\dots,D \qquad [w]
\end{aligned} \qquad (4)$$

Using a Lagrange multiplier we can solve this problem.[2]

4 Experiments and Results

4.1 Experiments with Grid world

In our first experiment, we used a 5×5 grid world as a toy example where the agent starts from the lower-left grid square and has to make its way to the upper-right grid square. In this experiment we mainly want to study the effect of using a different type of demonstrations and how successful our frame-work is at detecting any adversarial demonstrations.

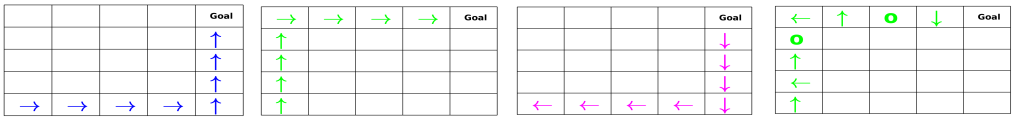
A reminder that our frame-work takes only the demonstrations as an input without any more information about its correctness and generates the policy and at the same time a w weight for each input demonstration. To best show how our algorithm is robust, we used three different types of demonstrations (Correct, adversarial, and random) as shown in Fig.1 .

As shown in Table 1 [3] , we can see the three different cases: (1) Using two correct demonstration the algorithm correctly assigns $w = 0.5$ for each demo and used both to generate the policy (accuracy

[2] A complete derivation and more details about the optimization algorithm can be found in Appendix B.

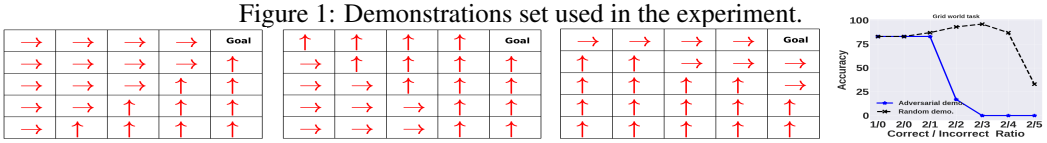
[3] Can be found in Appendix C.

= 100 %); (2) In the second case the algorithm assigns $w = 0.5$ to the two correct demonstrations and $w = 0.0$ to the adversarial demonstrations(accuracy= 83 %); (3) In the third case the algorithm assigns $w = 0.5$ to the two correct demonstrations and $w = 0.0$ to the random demonstrations (accuracy= 92 %). One last note in cases of using a random demonstrations the frame-work is able to detect those random demonstrations even if the number of correct demonstrations is less, that's because the entropy is a measurement of the randomness in the model, and the more random actions are taken the higher the entropy will be and it will be easier to detect as shown in Fig.2(d).



(a) Demo. 1 (Correct) (b) Demo. 2 (Correct) (c) Demo. 3 (Adversarial) (d) Demo. 4 (Random)

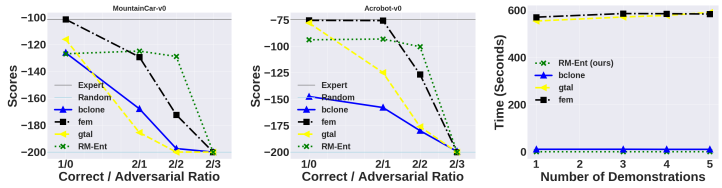
Figure 1: Demonstrations set used in the experiment.



(a) (b) (c) (d)

Figure 2: 2(a) is the result of using both of the correct demonstration as a mixture, 2(b) is the result of using correct demo. and an adversarial demo. , 2(c) is the result of using correct demo. and a random demo. , 2(d) is the accuracy using different correct/incorrect ration in case of random and adversarial demonstrations.

4.2 Experiments with OpenAI-Gym Simulator



(a) (b) (c)

Figure 3: Results of *Mountain-Car* and *Acrobot* experiments.

We run our algorithm on the classical control tasks *Mountain-Car* [14] and *Acrobot* [8] in the OpenAi-Gym simulator [4]. Both tasks have a continuous state space and discrete actions. Our main opponent is **BC** [3], we model π_{BC} using a neural network with parameter θ_{BC} and find these parameters using maximum-likelihood estimation such that $\theta_{BC} = \arg\max_\theta \prod_{(s,a)\in D} \pi_{BC}(a|s)$. Also, we compared our algorithm against one of the recent approach in IRL [11] with two different objective function; (1) Linear cost function from [1] (FEM); (2) Game-theoretic apprenticeship learning (GTAL): the algorithm of [11] using the cost function from [25].[4].

Fig. 3(a),3(b) shows the performance of different algorithms, under varying numbers of expert and adversarial demonstrations. We can see at the first point that RM-ENT is like BC as we use only correct demonstrations. However, starting from the second point we can see the power of our algorithm as it detects that we have an adversarial demonstration among the data set and remove it (set it's weight to zero) which will keep our accuracy unchanged. While other algorithms accuracy will decrease due to the adversarial demonstration. At the final point where we have more adversarial demonstration than the correct demonstrations, all the algorithms go to a random-like policy. We compared the time required to train each algorithm. As shown in Fig. 3(c) , RM-ENT requires much less time to converge, the reason for this is the use of neural network to train and run the opponent algorithms.

5 Conclusion and Future Work

In this work, we presented a novel frame-work that is able to automatically assign the proper weight for each of the given demonstrations and exclude the adversarial ones from the data-set. Our algorithm can achieve superior performance and sample efficiency than BC and IRL approaches in case of the presence of adversarial demonstrations. For future work, it would be enticing to use better optimization approach and extend the frame-work to handle continuous action space.

[4]More details about the experiment parameter and number of samples can be found in Appendix C.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, page 1. ACM, 2004.

[2] Shun-ichi Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

[3] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

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

[5] Sonia Chernova and Andrea L Thomaz. Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(3):1–121, 2014.

[6] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

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

[8] Alborz Geramifard, Christoph Dann, Robert H Klein, William Dabney, and Jonathan P How. Rlpy: a value-function-based reinforcement learning framework for education and research. 2015.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[10] Daniel H Grollman and Aude Billard. Donut as i do: Learning from failed demonstrations. In *2011 IEEE International Conference on Robotics and Automation*, pages 3804–3809. IEEE, 2011.

[11] Jonathan Ho, Jayesh Gupta, and Stefano Ermon. Model-free imitation learning with policy optimization. In *International Conference on Machine Learning*, pages 2760–2769, 2016.

[12] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

[13] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. *arXiv preprint arXiv:1809.02925*, 2018.

[14] Andrew William Moore. Efficient memory-based learning for robot control. 1990.

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

[16] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[17] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018.

[18] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

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

[20] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[21] Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, 1998.

[22] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[23] Kyriacos Shiarlis, Joao Messias, and SA Whiteson. Inverse reinforcement learning from failure. 2016.

[24] Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. In *Proceedings of the 25th international conference on Machine learning*, pages 1032–1039. ACM, 2008.

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

[26] Umar Syed and Robert E Schapire. A reduction from apprenticeship learning to classification. In *Advances in neural information processing systems*, pages 2253–2261, 2010.

[27] Jiangchuan Zheng, Siyuan Liu, and Lionel M Ni. Robust bayesian inverse reinforcement learning with sparse behavior noise. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[28] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

A Appendix

A.1 Dual Problem Derivation

Starting from the primal problem:

$$\begin{aligned} \max_{\pi \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}} \quad & H(\pi) \equiv -\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \tilde{p}(s)\pi(a|s)\log\pi(a|s) \\ \text{s.t.} \quad & \mathbb{E}_{\tilde{\pi}}[f_i] - \mathbb{E}_{\pi}[f_i] = 0 \quad i = 1,\dots,n \\ & \sum_{a\in\mathcal{A}} \pi(a|s) - 1 = 0 \quad \forall\, s \in \mathcal{S} \end{aligned} \tag{5}$$

To derive the dual problem we will use the Lagrange method for convex optimization problems.

$$\Lambda(\pi,\lambda,\mu) \equiv H(\pi) + \sum_{i=1}^{N}\lambda_i\Big(\mathbb{E}_{\pi}[f_i] - \mathbb{E}_{\tilde{\pi}}[f_i]\Big) + \sum_{s\in\mathcal{S}}\tilde{p}(s)\mu_s\Big(\sum_{a\in\mathcal{A}}\pi(a|s) - 1\Big) \tag{6}$$

Where λ_i, μ_s are the Lagrangian's multiplier corresponding to each constraint.

$$\Lambda(\pi,\lambda,\mu) \equiv -\sum_{s\in\mathcal{S}}\tilde{p}(s)\sum_{a\in\mathcal{A}}\pi(a\mid s)\log\pi(a\mid s) + \sum_{i=1}^{N}\lambda_i\Big(\mathbb{E}_{\pi}[f_i] - \mathbb{E}_{\tilde{\pi}}[f_i]\Big) + \sum_{s\in\mathcal{S}}\tilde{p}(s)\mu_s\Big(\sum_{a\in\mathcal{A}}\pi(a|s) - 1\Big) \tag{7}$$

By Differentiating the Lagrangian with respect to primal variables $p(s|a)$ and letting them to be zero, we obtain:

$$\frac{\partial\Lambda}{\partial\pi(a|s)} = -\sum_{s\in\mathcal{S}}\tilde{p}(s)\Big(1 + \sum_{a\in\mathcal{A}}\log\pi(a|s)\Big) + \sum_{i=1}^{N}\lambda_i\Big(\sum_{s\in\mathcal{S}}\tilde{p}(s)\sum_{a\in\mathcal{A}}f(s,a)\Big) + \sum_{s\in\mathcal{S}}\tilde{p}(s)\mu_s \tag{8}$$

$$-\sum_{s\in\mathcal{S}}\tilde{p}(s)\Big(1 + \sum_{a\in\mathcal{A}}\log\pi(a|s)\Big) + \sum_{i=1}^{N}\lambda_i\Big(\sum_{s\in\mathcal{S}}\tilde{p}(s)\sum_{a\in\mathcal{A}}f(s,a)\Big) + \sum_{s\in\mathcal{S}}\tilde{p}(s)\mu_s = 0 \tag{9}$$

$$\sum_{s\in\mathcal{S}}\tilde{p}(s)\Bigg(-1 - \sum_{a\in\mathcal{A}}\log\pi(a|s) + \sum_{i=1}^{N}\lambda_i\Big(\sum_{a\in\mathcal{A}}f(s,a)\Big) + \mu_s\Bigg) = 0 \tag{10}$$

Assuming $\tilde{p}(s) \neq 0$,

$$\log\pi(a|s) = \sum_{i=1}^{N}\lambda_i\Big(f_i(s,a)\Big) + \mu_s - 1 \tag{11}$$

$$\pi(a|s) = \exp\Bigg(\sum_{i=1}^{N}\lambda_i\Big(f_i(s,a)\Big)\Bigg)\cdot\exp\Big(\mu_s - 1\Big) \tag{12}$$

Since $\sum_{a\in\mathcal{A}}\pi(a|s) = 1$

$$\sum_{a\in\mathcal{A}}\exp\Bigg(\sum_{i=1}^{N}\lambda_i\Big(f_i(s,a)\Big)\Bigg)\cdot\exp\Big(\mu_s - 1\Big) = 1 \tag{13}$$

$$\frac{1}{\sum_{a\in\mathcal{A}} \exp\Big(\sum_{i=1}^{N} \lambda_i\Big(f_i(s,a)\Big)\Big)} = \exp\Big(\mu_s - 1\Big) = (z_\lambda(s))^{-1} \tag{14}$$

By substituting in Eq 12 we will get.

$$\pi^*(a|s) = (z_\lambda(s))^{-1} \cdot \exp\Big(\sum_{i=1}^{N} \lambda_i\Big(f_i(s,a)\Big)\Big) \tag{15}$$

Finally, the dual problem will be:

$$-\Big\{\max_{\lambda}\ \Lambda(\lambda) \equiv -\sum_{s\in\mathcal{S}} \tilde{p}(s)\log z_\lambda(s) + \sum_{i=1}^{N}\lambda_i \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \tilde{\pi}(s,a) f(s,a)\Big\} \tag{16}$$

B Appendix

B.1 Dual Problem of Robust Maximum Behavior Cloning

We will start from Eq. 16 and build upon it. As we mentioned in the main text we will introduce the w weight as part of our model.

$$\tilde{p}_w(s) = \frac{1}{M}\sum_{d=1}^{D} w_D \tilde{p}(s|d) \tag{17a}$$

$$\tilde{\pi}_w(s,a) = \frac{1}{M}\sum_{d=1}^{D} w_D \tilde{\pi}(s,a|d) \tag{17b}$$

By substituting in Eq 16 we will get.

$$\begin{aligned}\min_{w}\quad -\Big\{\max_{\lambda}\quad & \Lambda(\lambda) \equiv \frac{1}{M}\sum_{d=1}^{N} w_d\Big(-\sum_{s\in\mathcal{S}} \tilde{\pi}_w(s|d)\log z_\lambda(s) + \sum_{i=1}^{N}\lambda_i\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\tilde{\pi}_w(s,a|d) f(s,a)\Big)\Big\} \\ \text{s.t.}\quad & \sum_{d=1}^{D} w_d = M \\ & w_d \geq 0 \quad \forall d \in \mathcal{D} = 1....D \\ & w_d \leq 1 \quad \forall d \in \mathcal{D} = 1,\ldots,D\end{aligned} \tag{18}$$

For simplification let's assume:

$$a_d = \sum_{s\in\mathcal{S}} \tilde{\pi}(s|d)\log z_\lambda(s) \tag{19}$$

$$b_d = \sum_{i=1}^{N}\lambda_i \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \tilde{\pi}(s,a|d) f(s,a) \tag{20}$$

$$c_d = b_d - a_d \quad \forall d \in D = 1,\ldots,D \tag{21}$$

$$\begin{aligned} \min_{w} \quad & -\Big\{\max_{\lambda} \Lambda(\lambda) \equiv \frac{1}{M}\sum_{d=1}^{D} w_d c_d\Big\} \\ \text{s.t.} \quad & \sum_{d=1}^{N} w_d = M \\ & w_d \leq 1 \quad \forall d \in \mathcal{D} \\ & w_d \geq 0 \quad \forall d \in \mathcal{D} \end{aligned} \tag{22}$$

By moving the negative sign to inside we will reach our final optimization problem.

$$\begin{aligned} \min_{\lambda, w} \quad & \Lambda(\lambda) \equiv -\frac{1}{M}\sum_{d=1}^{D} w_d c_d \\ \text{s.t.} \quad & \sum_{d=1}^{D} w_d = M \\ & w_d \leq 1 \quad \forall d \in D = 1....D \\ & w_d \geq 0 \quad \forall d \in D = 1....D \end{aligned} \tag{23}$$

From the last equation, we can see its a non-convex problem, we used Sequential Quadratic Programming (SQP) approach to solve this problem, the basic SQP algorithm is described in chapter 18 of Nocedal and Wright [16].

SQP approach allows you to closely mimic Newton's method for constrained optimization just as is done for unconstrained optimization. At each major iteration, an approximation is made of the Hessian of the Lagrangian function using a quasi-Newton updating method. This is then used to generate a Quadratic Programming (QP) subproblem whose solution is used to form a search direction for a line search procedure. we leveraged the function implementation in Matlab and used it to solve our problem. [5]

C Appendix

C.1 Grid World Experiment

Table 1: Results of grid world

Experiment	Demo. number	Demo. Type	Accuracy	Weights	Cor. / Adv.
Mixture of correct	demo._1, Fig.1(a)	"correct"	100 % Fig.2(a)	0.5	2/0
	demo._2, Fig.1(b)	"correct"		0.5	
Correct & Adversarial	demo._1, Fig.1(a)	"correct"	83 % Fig.2(b)	0.5	2/1
	demo._2, Fig.1(a)	"correct"		0.5	
	demo._3, Fig.1(c)	"adversarial"		0.0	
Correct & Random	demo._1, Fig.1(b)	"correct"	92 % Fig.2(c)	0.5	2/3
	demo._2, Fig.1(b)	"correct"		0.5	
	demo._3, Fig.1(d)	"random"		0.0	
	demo._4, Fig.1(d)	"random"		0.0	
	demo._5, Fig.1(d)	"random"		0.0	

C.2 Classical Control Experiments in OpenAI-Gym Simulator Details

The expert data was generated using TRPO [22] on the true cost functions. For the adversarial demonstrations, we simply manipulated the actions of the expert data. For example, in the mountain

[5] https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html#bsgppl4

Table 2: Parameter for FEM and GTAL

Task	Training iterations	State-action pairs per iteration
Mountain Car	300	5000
Acrobot	300	5000

car, we had two actions 0,1. If the expert data was taking action 0 with a specific observation we replaced it with action 1 and vice versa. The idea behind that is to generate an adversarial demonstration that tries to fool the algorithm.

For a fair comparison, we used the same experimental settings as in [11], including the exact neural network architectures for the policies and the optimizer parameters for TRPO [22] for all of the algorithms except ours which do not use any neural network.

The amount of environment interaction used for FEM and GTAL is shown in Table 2 . A reminder that BC and RM-ENT do not use any more samples during the training.