Robust Probabilistic Imitation Learning

Brendan Crowe

December 2020

Abstract

As Imitation Learning (IL) becomes a more popular area of research and its applications reach real world problems, the necessity to deal with adversarial data in a structured way grows. Most approaches as of now assume the data is perfect, no adversary. However, in real world applications this assumption can be detrimental. Being able to detect and remove adversarial demonstrations is paramount for the success of IL. This paper proposes a probabilistic approach to autonomously detect and re-weight adversarial demonstrations, as it learns a policy from the expert demonstrations. The problem posed in this work can then be elegantly solved using an Expectation Maximization (EM)-esque algorithm. The proposed method is statistically robust to adversarial demonstrations and thus will be titled Robust Probabilistic Imitation Learning (R-PIL). Furthermore, being probabilistic in nature, this could be used as a general method for re-weighting demonstrations in other frameworks.

1 Introduction

Imitation Learning (IL) or learning a task from demonstrations, has many theoretical applications in robotics, control systems, etc. In recent years, IL has achieve great results in simulations and controlled experiments [**ho2016generative**] [**ramachandran2007bayesian**]. An assumption that many IL frameworks make today is that the demonstrations provided are completing the task in an optimal way, and that they are correct, not failed or corrupted. While this assumption is easy to verify in simulations and laboratory experiments, that may not be possible in the real world. When attempting to teach an agent (a robot or other device) to learn a task, the exact dynamics or optimal solution, may be unknown to the modeler. In this situation, it could be difficult or time consuming to determine if any of the demonstrations are following a non-optimal or downright incorrect strategy. An adversary in this context, is any agent (human or other) that is not following an optimal or safe policy for the task, where a policy being the way that one completes the task. These adversarial demonstrations can be detrimental to IL not only because they could prevent learning of the task, but because they could cause an agent to learn an unsafe way to approach the task.

Let us examine an example. We want to teach a self driving car by have experts (humans) show the car how to drive. Humans would drive the car and sensor embedded in the car would collect various features that could be used to train an autonomous agent. In this case, we could have several kinds of drivers that would be adversarial. There could be someone who is a poor drivers, reckless drives, or simply the sensors in the car could be corrupted. In the case of a self driving car, the car may be able to learn to drive despite adversarial demonstration, but it might learn unsafe behavior as a result e.g. speeding, failure to stop at stop signs, passing on the right, illegal u-turns, etc.

In this work, we will discuss IL in the context of logistic regression, which can be a general way of solving this problem. More pointedly, we will discuss a probabilistic approach to detect and re-weight demonstrations coming from the adversary. Finally, we will discus how this can be generalized to other models.

2 Imitation Learning

The goal of IL is to learn a mapping of states s to actions a in the form of a policy $\widetilde{\pi}$ from expert demonstrations. We assume that we have a set of expert demonstrations $\mathcal{D}$ such that $\mathcal{D} = \{d_m : (a_n, s_n)\}$ for $n = 1, \ldots, N_d$ and $m = 1, \ldots, M$ where $s \in \mathcal{S}$ $a \in \mathcal{A}$ the state and action spaces for the task. These expert demonstrations are generated from and expert policy π^*.

3 Related work

The inspiration for this work came from a related approach to IL that dealt with adversarial demonstrations. Robust Maximum Entropy Behavior Cloning (RM-ENT) [**hussein˙crowe˙petrik˙begum˙2020**] focuses on a robust Maximum Entropy based method that can autonomously detect adversarial demonstrations and remove them from the data set. While trying to bring a theoretical grounding to the results shown by the method a connection between the dual of the original problem posed in the paper, and the maximum likelihood estimate for multinomial logistic regression was found.

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

$$\equiv$$

$$\arg\max_{\lambda}\ \ell\left(\lambda|\mathcal{A},\mathcal{S}\right) = -\sum_{n=1}^{N}\widetilde{p}(s_n)\log(z_\lambda(i)) + \sum_{n=1}^{N}\sum_{k=1}^{K}\widetilde{p}(s_n,a_k)\sum_{j=1}^{J} f_{ij}\lambda_{jk}\text{[\textbf{hussein˙crowe˙petrik˙begum˙2020}]} \tag{1}$$

This lead to further investigation if there was an equivalent method to version of RM-ENT that can detect adversaries.

$$\begin{aligned}&\min_{w\in\mathbb{R}^D,\lambda\in\mathbb{R}^N}\ \Lambda(\lambda,w) \equiv -\frac{1}{M}\sum_{d=1}^{D} w_d\Big(-\sum_{s\in\mathcal{S}}\tilde{p}(s,d)\log z_\lambda(s) + \sum_{i=1}^{N}\lambda_i\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\tilde{\pi}(a|s,d)f(s,a)\Big)\\ &\text{s. t.}\quad \sum_{d=1}^{D} w_d = M, \quad w_d \geq 0 \quad \forall d\in\mathcal{D}, \quad w_d \leq 1 \quad \forall d \in \mathcal{D}\end{aligned} \tag{2}$$

Both methods attempt to solve the same problem, but model the adversary slightly different. While RM-ENT solves a constrained optimization problem that enforces a number of correct demonstrations, R-PIL re-weights the demonstration accordion the likelihood that said demonstrations belongs to the expert model. This is a slightly different approach semantically and mathematically but achieves similar results.

4 Multinomial Logistic Regression

A simple and powerful tool for linear classification is multinomial logistic regression or softmax regression. As we saw in the last section, there is a connection between RM-ENT and logistic regression. In logistic regression the parameters are maximized with respect to the likelihood function as such it is a maximum likelihood estimator (mle). To gain an understanding for logistic regression in the IL context, we will define the probability of an action given in corresponding state as so:

$$p_\lambda(a_k|s_n) = \frac{e^{\sum_{j=1}^{J} f_{nj}\lambda_{jk}}}{\sum_{a' \in \mathcal{A}} e^{\sum_{j=1}^{J} f_{nj}\lambda'_j}} \tag{3}$$

This assumes the actions follow a multinomial distribution and are thus finite. From this, we can derive our likelihood as the product over our training examples and all possible actions.

$$\begin{aligned} L(\lambda|\mathcal{D}) &= \prod_{m=1}^{M}\prod_{i=1}^{N_d}\prod_{k=1}^{K} p_\lambda(a_k|s_i)^{\widetilde{p}(s_i,a_k|d_m)} \\ \widetilde{p}(s_i,a_k|d_m) &\in \{0,1\} \end{aligned} \tag{4}$$

Here, K represents the number of actions in the action space, and $\widetilde{p}(s_i, a_k|d_m)$ is the probability of observing a state-action pair in demonstration d_m. The following is the corresponding log likelihood:

$$\ell(\lambda|\mathcal{D}) = \sum_{m=1}^{M}\sum_{n=1}^{N_d}\sum_{k=1}^{K} \widetilde{p}(s_n, a_k|d_m)\log(p_\lambda(a_k|s_n)) \tag{5}$$

5 Modeling the Adversary

To be able to detect the adversarial demonstrations within the demonstrations set, we need a way of expressing the adversary in the likelihood. A natural way of doing this is expressing our likelihood as a mixture of the expert and the adversary:

$$\prod_{m=1}^{M}\prod_{n=1}^{N_d} (p(a_n|\text{Expert}) + p(a_n|\text{Adversary})) \tag{6}$$

This leaves us with a new question: what is the probability of the correct action given the adversary? There are three fairly simple and reasonable ways of doing this. We will denote the adversarial model as ψ

1. $p(a_n|\text{Adversary}) = 1$
   This is an easy way to model the adversary but would require additional constraints during optimization to stop all demonstrations being marked as adversarial.

2. $p(a_n|\text{Adversary}) = p_\psi(a_k|s_i)^{\widetilde{p}(s_i,a_k|d_m)}$
   Model the adversary as its own logistic model with its own learnable parameters. This would only be able to learn a separate model for the expert and adversary if both of the distributions are distinct. Otherwise, there is nothing constraining the models from learning equally bad policies.

3. $p(a_n|\text{Adversary}) = 1 - p_\lambda(a_k|s_i)^{\widetilde{p}(s_i,a_k|d_m)}$
   The probability of the adversary is the compliment of the probability for the expert.

Option 3 is straightforward to design and theoretically sound. Using the compliment probability of the expert to model, the adversary works on the assumption that we have two distinct sets of data in $\mathcal{D}$. We have demonstrations generated completely by the expert, and demonstrations generated completely by the adversary. With this assumption we can say that any data point not modeled correctly by the expert belongs to the adversary, hence the compliment.

However, this model still ignores the fact that we do not actually know which demonstrations belong to the the expert or adversary. This is a critical component of detecting the adversarial demonstration. To model this we will introduce a latent variable Z a binomial random variable that represents the two demonstrators we have. We will further introduce the two settings for $Z : \Lambda\,\text{Expert}, \quad \Psi\,\text{Adversary}$. What follows is a definition of what is at the core of this work.

$$p(a_n|\Psi) = 1 - p(a_n|\Lambda) \tag{7}$$

Not only does this give us a model that can detect adversarial demonstration and learn a policy, but it also allows the modeler to express the amount of the demonstrations they believe to be correct in the form of the prior distribution over Z. Now we can express our full model as so:

$$\begin{aligned} L(\lambda|\mathcal{D}) &= \prod_{m=1}^{M}\prod_{i=1}^{N_d}\prod_{k=1}^{K}\left(\left(p(\lambda)p_\lambda(a_k|s_i)^{\widetilde{p}(s_i,a_k|d_m)}\right) - \left(p(\psi)(1 - p_\lambda(a_k|s_i)^{\widetilde{p}(s_i,a_k|d_m)})\right)\right) \\ p(\psi) &= 1 - p(\lambda) \end{aligned} \tag{8}$$

$$\begin{aligned} L(\lambda|\mathcal{D}) &= \prod_{m=1}^{M}\prod_{i=1}^{N_d}\prod_{k=1}^{K}\sum_{Z}\left(p(z)p_z(a_k|s_i)^{\widetilde{p}(s_i,a_k|d_m)}\right) \\ p_\psi(a_k|s_i)^{\widetilde{p}(s_i,a_k|d_m)} &= 1 - p_\lambda(a_k|s_i)^{\widetilde{p}(s_i,a_k|d_m)} \end{aligned} \tag{9}$$

While this new likelihood models the adversary, it poses a problem for optimization as can be seen from the expression of the log likelihood:

$$\ell(\lambda|\mathcal{D}) = \sum_{m=1}^{M}\sum_{i=1}^{N_d}\sum_{k=1}^{K}\log\left(\sum_{Z}\left(p(z)p_z(a_k|s_i)^{\widetilde{p}(s_i,a_k|d_m)}\right)\right) \tag{10}$$

As can be seen, the summation over our latent variable Z causes our likelihood to become non-convex.

6 Expectation Maximization

In the last section, we derived a likelihood that models the problems as a mixture of the expert and adversary. This causes a non-convexity which makes optimizing our parameters difficult. Luckily for us, there is Expectation Maximization (EM), a powerful statistical algorithm that deals with latent variables. "An elegant and powerful method for finding maximum likelihood solutions for models with latent variables is called the expectation-maximization algorithm, or EM algorithm" [**10.5555/1162264**]. EM uses Jensen's inequality to form a lower bound for the log likelihood. This lower bound approximation is convex and therefore easy to solve. The original likelihood can be solved by iterative approximation of the likelihood with its lower bound, and then by maximizing that function. EM in general:

1. Initialize parameters θ^{old}
2. E Step: Evaluate $p(Z|X, \theta^{\text{old}})$
3. M Step: Evaluate $\theta^{\text{new}} = \arg\max_{\theta} Q(\theta, \theta^{\text{old}})$
   Where $Q(\theta, \theta^{\text{old}}) = \sum_Z p(Z|X, \theta^{\text{old}}) \log(p(X, Z|\theta))$
4. Check for convergence
   If not converged $\theta^{\text{old}} \leftarrow \theta^{\text{new}}$ and return to step 2
   [**10.1145/3054912**]

We then build upon this concept to construct an iterative solution to our problem.

6.1 Expectation Step

We need to express the probability of Z given a demonstration d_m and our initial parameters λ^{old}. Using Bayes' theorem, $p(Z|d_m, \lambda^{\text{old}}) = \frac{p(Z)p(d_m|Z)}{p(d_m)}$, since d_m is already observed we will take its probability to be 1. Thus, $p(Z|d_m, \lambda^{\text{old}}) = p(Z)p(d_m|Z)$. Keeping in mind (7) we derive the following expression:

$$\begin{aligned} p(z|d_m, \lambda^{\text{old}}) &= \frac{p(z)p(d_m|z)}{\sum_{z' \in Z} (p(z')p(d_m|z'))} \quad \text{for} z \in \{\Lambda, \Psi\} \\ p(d_m|\Lambda) &= \prod_{n=1}^{N_d} p_\lambda(a_n|s_n) \\ p(d_m|\Psi) &= \prod_{n=1}^{N_d} (1 - p_\lambda(a_n|s_n)) \end{aligned} \tag{11}$$

6.2 Maximization Step

Now we need to construct our lower bounding surrogate function $Q(\lambda, \lambda^{\text{old}})$:

$$\begin{aligned} Q(\lambda, \lambda^{\text{old}}) &= \sum_{z \in Z} \Big(p(z|d_m, \lambda^{\text{old}}) \log(p(\mathcal{D}, z|\lambda)) \Big) \\ &= \sum_{z \in Z} \Big(p(z|d_m, \lambda^{\text{old}}) \ell(\lambda|\mathcal{D}, z) \Big) \\ &= \sum_{m=1}^{M} \sum_{i=1}^{N_d} \sum_{k=1}^{K} \sum_{z \in Z} \Big((\log(p(z)) + \widetilde{p}(s_i, a_k|d_m) p_z(a_k|s_i))\, p(z|d_m, \lambda^{\text{old}}) \Big) \end{aligned} \tag{12}$$

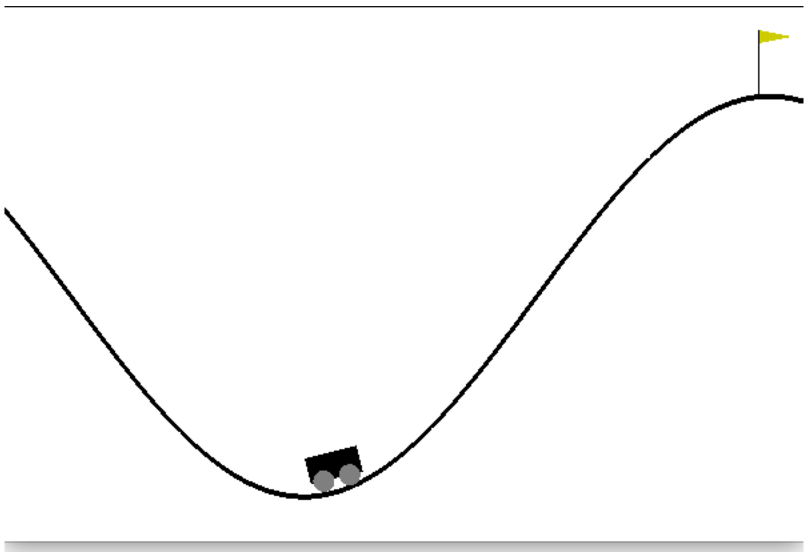
This surrogate function is to be convex and can be easily and efficiently optimized by several convex optimization methods. For this work, we leveraged the SciPy [**2020SciPy-NMeth**] implementation of Limited Memory Broyden–Fletcher–Goldfarb–Shanno algorithm L-BFGS-B. At times, for certain programming conveniences, we used Scikit-Learn [**scikit-learn**] as a wrapper to interact with SciPy.

7 Results

To show empirical results using this method we uses OpenAI-gym [**brockman2016openai**], an open source project for Reinforcement Learning. This allows us to assess our model's performance within the environment. We we can generate expert data for several different environments. For this work we will use the Mountain Car, Acrobot, and Lunar Lander environments [**brockman2016openai**]. We will examine two kinds of adversary, corrupted data and poor expert. We will use basic multinomial logistic regression (9) as baseline to evaluate the performance of the model.

7.1 Mountain Car

"A car is on a one-dimensional track, positioned between two "mountains". The goal is to drive up the mountain on the right; however, the car's engine is not strong enough to scale the mountain in a single pass. Therefore, the only way to succeed is to drive back and forth to build up momentum."[**brockman2016openai**]



This is a simple environment and is a great example to start with. It has a small state space, only position, and velocity of the car, which allows us to visualize the expert and learned policies. Below are some of the results.

Table 1: Mountain Car Corrupted Data

Metric	Average Reward
Expert	-103.9 ± 6.04
R-PIL	-104.45 ± 8.50
LogReg	-132.0 ± 14.36

Table 2: Mountain Car Random Policy

Metric	Average Reward
Expert	-103.9 ± 6.04
R-PIL	-106.25 ± 6.00
LogReg	-119.5 ± 1.75

7.2 Lunar Lander

"Landing pad is always at coordinates (0,0). Coordinates are the first two numbers in state vector. Reward for moving from the top of the screen to landing pad and zero speed is about 100..140 points. If lander moves away from landing pad it loses reward back. Episode finishes if the lander crashes or comes to rest, receiving additional -100 or +100 points. Each leg ground contact is +10. Firing main engine is -0.3 points each frame. Solved is 200 points. Landing outside landing pad is possible. Fuel is infinite, so an agent can learn to fly and then land on its first attempt. Four discrete actions available: do nothing, fire left orientation engine, fire main engine, fire right orientation engine." [**brockman2016openai**]

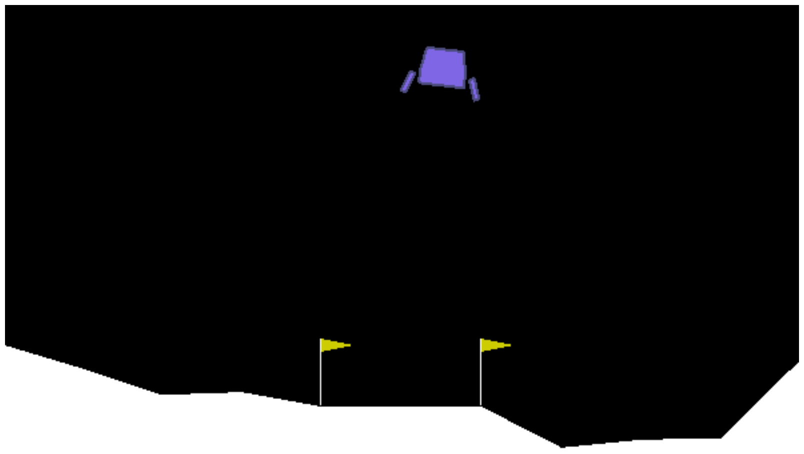


Table 3: Lunar Lander Corrupted Data

Metric	Average Reward
Expert	254.45 ± 57.09
R-PIL	195.06 ± 64.45
LogReg	-22.24 ± 48.51

Table 4: Lunar Lander Random Policy

Metric	Average Reward
Expert	254.45 ± 57.09
R-PIL	226.49 ± 49.52
LogReg	-342.64 ± 35.53

7.3 Acrobot

"The acrobot system includes two joints and two links, where the joint between the two links is actuated. Initially, the links are hanging downwards, and the goal is to swing the end of the lower link up to a given height." [**brockman2016openai**]

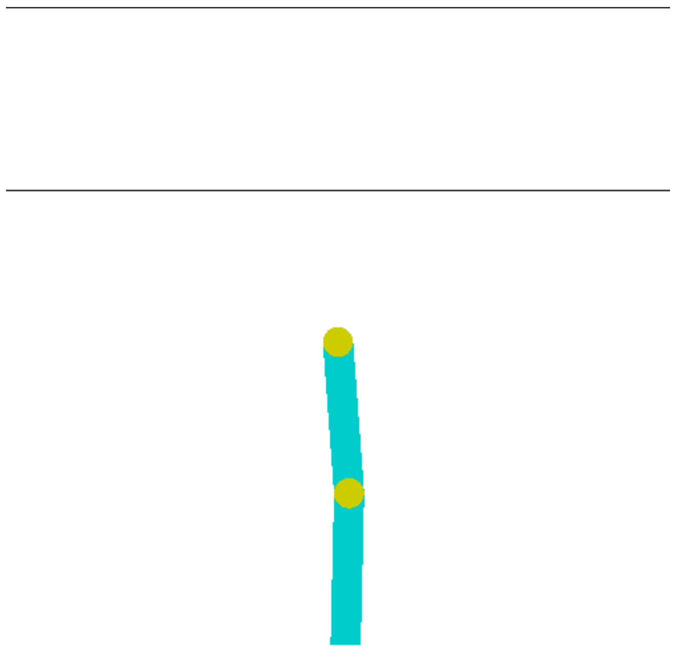


Table 5: Acrobot Corrupted Data

Metric	Average Reward
Expert	-80.65 ± 3.49
R-PIL	-102.7 ± 22.62
LogReg	-102.05± 31.67

Table 6: Acrobot Random Policy

Metric	Average Reward
Expert	-80.65 ± 3.49
R-PIL	-92.85 ± 23.81
LogReg	-397.65 ± 85.18

8 Future Work

Going forward with the project, it would be interesting to apply this to other frameworks. As the core of the contribution of this paper is in dealing with adversarial demonstrations through probabilistic re-weighting, this can be generalized to a number of different models. It would also be interesting to expand this concept to non-discrete action spaces. If these extensions are possible this method would be of great interest in the IL community.

9 Citations

10 Apendix

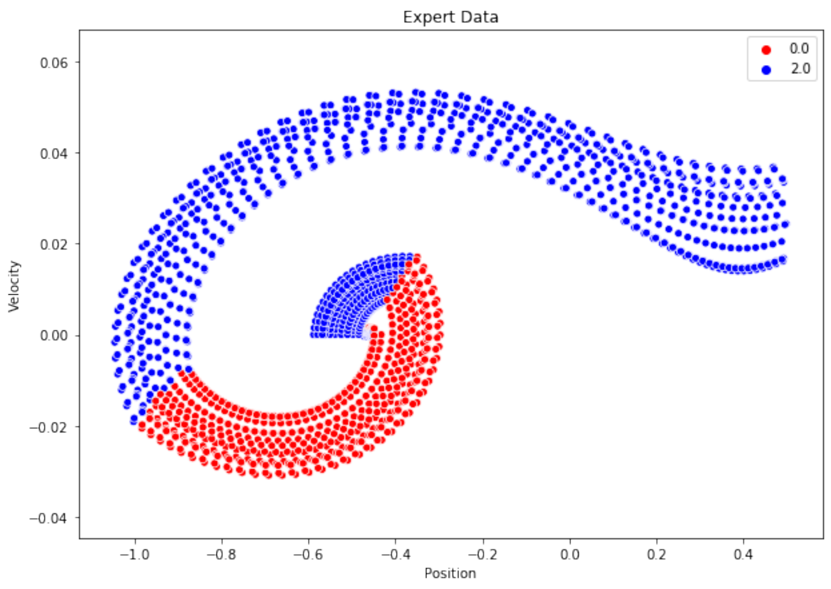


Figure 1: 20 Expert Demonstrations

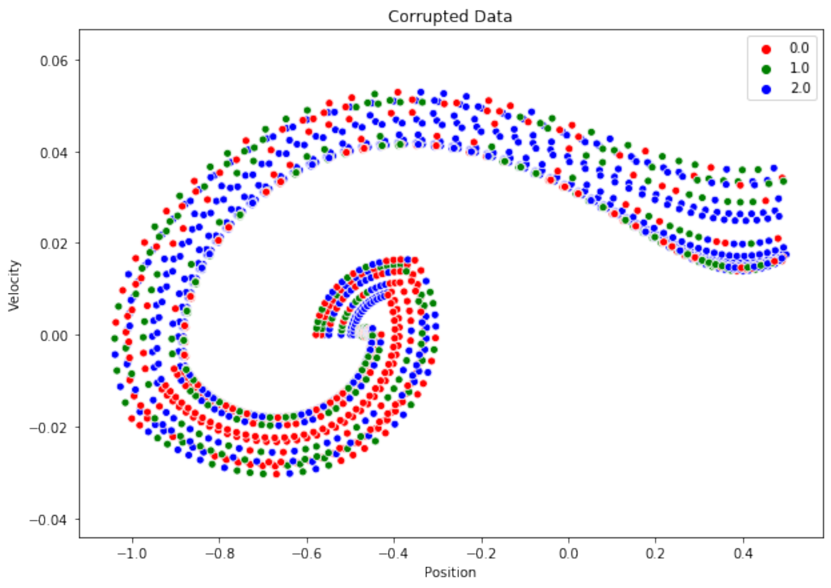


Figure 2: 10 Expert demonstrations, 10 Corrupted Demonstrations

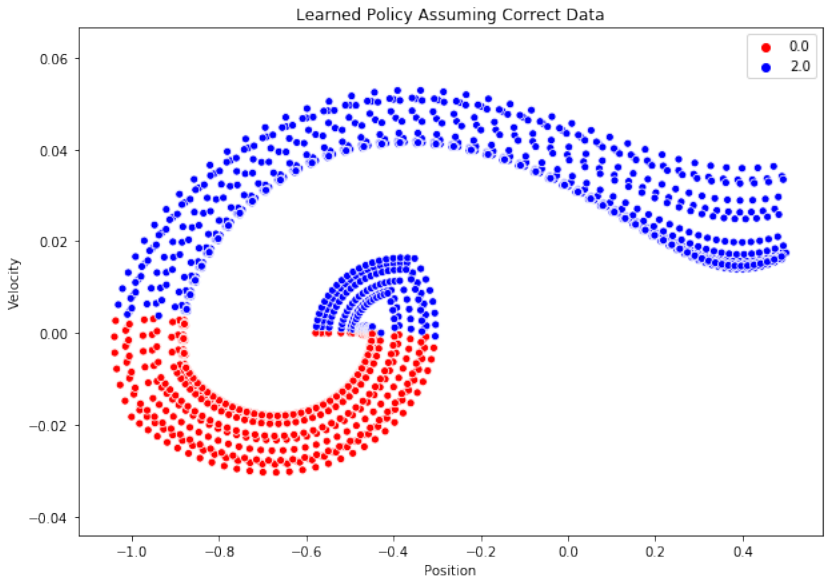


Figure 3: Learned policy assuming demonstrations are correct

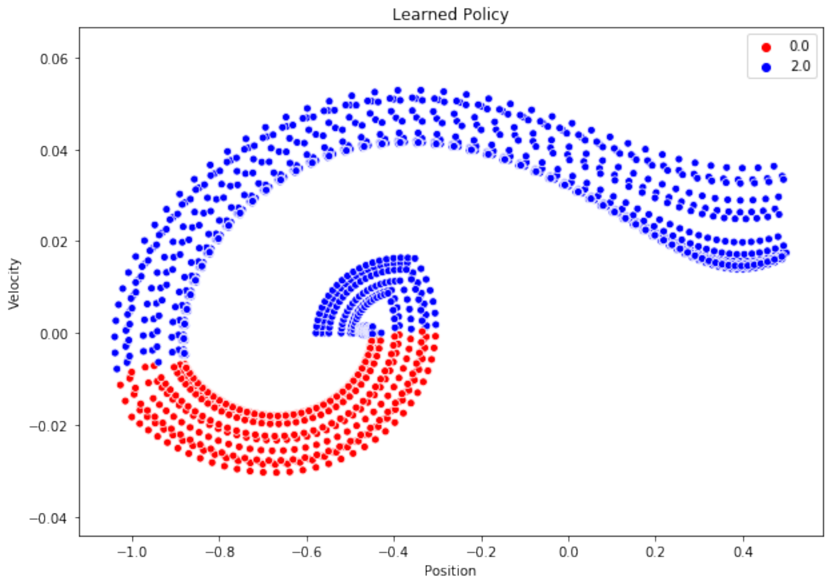


Figure 4: Learned policy R-PIL

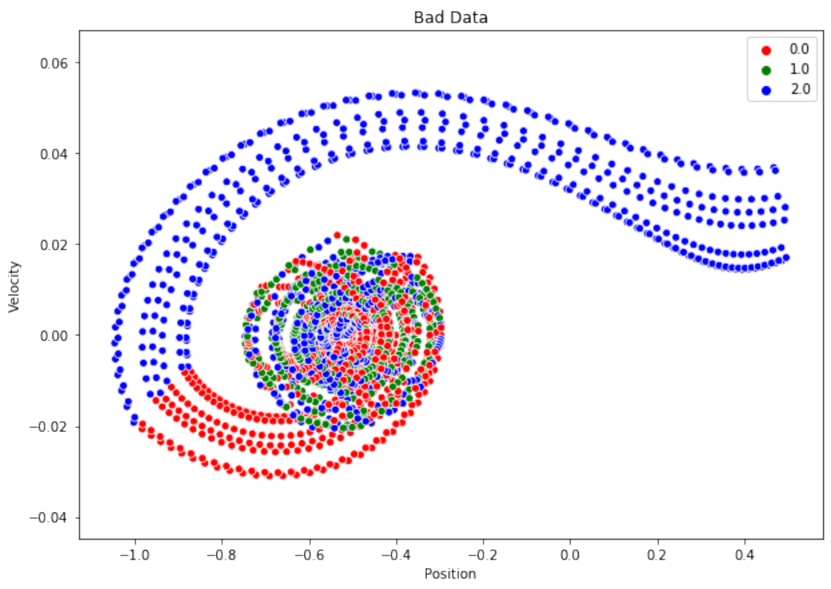


Figure 5: 10 expert demonstrations, 10 demonstrations following a random policy

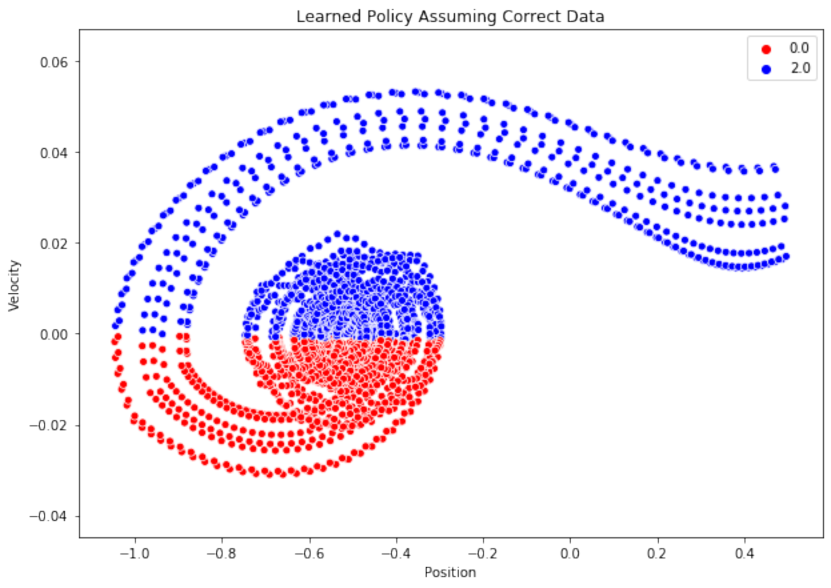


Figure 6: Learned policy assume demonstrations are correct

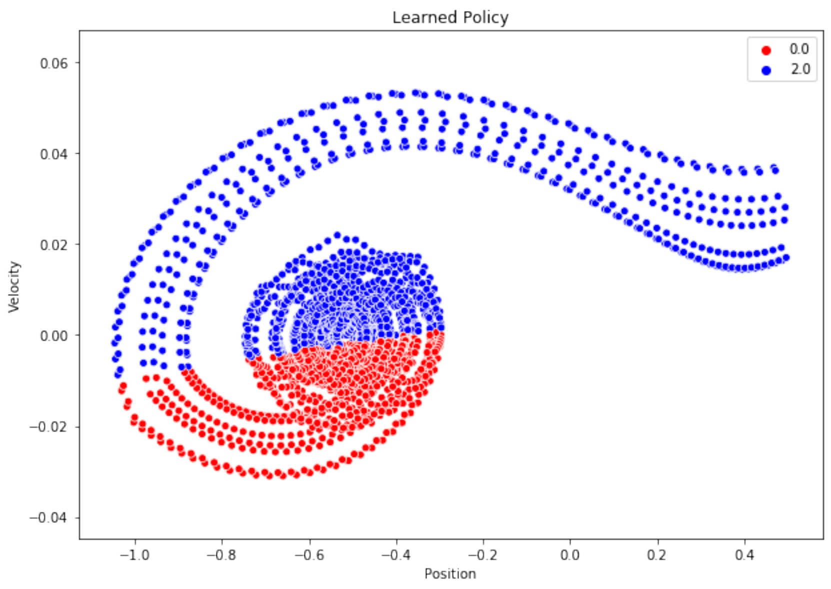


Figure 7: Learned policy R-PIL