Robust Probabilistic Imitation Learning

Connections to Robust Maximum Entropy Behavior Cloning

Brendan Crowe

University of New Hampshire
Department of Mathematics and Statistics
Department of Computer Science

TABLE OF CONTENTS

Imitation Learning

General Definition: Imitation learning (IL) techniques aim to mimic human behavior in a given task. An agent (a learning machine) is trained to perform a task from demonstrations by learning a mapping between observations and actions.[3]

- Teaching robots new tasks
- Domain experts can train agents without knowledge of machine learning
- Less explicit programming

Framework

- States
- Actions
- Transitions
- Rewards
- Policy: solution
- Expert
- Demonstrations

Imitation Learning Paradigms

- **Behavior Cloning (BC)**: Methods learn a mapping from states to actions as a supervised learning problem [5]

- **Feature Expectation Matching (FEM)** is employed to generate a policy; however there is non uniqueness

- **Inverse Reinforcement Learning (IRL)**: Attempt to recover the reward function the agent is trying to optimize. Then optimize that reward function.

- **Generative Adversarial Imitation Learning (GAIL)** has achieved great results [2]

Motivation: Adversarial Demonstrations

- Assumption that demonstrations are correct
- Sample inefficient
- Model dependent
- Use of simulators

My Approach

1. Model an adversary using likelihoods
2. Solve non convex optimization using Expectation Maximization
3. Remove demonstrations from the training set

Behavioral Cloning as Logistic Regression

- $a_k \in \mathcal{A} \quad k = 1, \ldots, K$
- $s_n \in \mathcal{S} \quad n = 1, \ldots, N_d$
- N_d for $d_m \in \mathcal{D} \quad d = 1, \ldots, M$
- Features $f_j \quad j = 1, \ldots, J$ that define a state
- Learnable parameters λ

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

Maximum Likelihood

Now we want to maximize the likelihood of the parameter given our expert data

$$L(\lambda|\mathcal{D}) = \prod_{m=1}^{M}\prod_{i=1}^{N_d}\prod_{k=1}^{A} p_\lambda(a_k|s_i)^{\widetilde{p}(s_i,a_k|d_m)}$$

Where A is the size of the action space and $\widetilde{p}(s_i, a_k|d_m)$ observed probability distribution of state-action pairs for a given demonstration. In real data $\widetilde{p}(s_i, a_k|d_m) \in \{0, 1\}$, whether that state action pair was observed.

Connections to RM-ENT BC

- Robust Maximum Entropy Behavior Cloning (RM-ENT BC) [4]
- Connection between the dual and logistic regression

How do we model the adversary?

$$L(\lambda|\mathcal{D}) = \prod_{m=1}^{M}\prod_{n=1}^{N_d}(p(a_n|\text{expert}) + p(a_n|\text{adversary}))$$

This raises a new problem, what is the probability of an action given the expert or adversary?

How do we model the adversary?

$$L(\lambda|\mathcal{D}) = \prod_{m=1}^{M}\prod_{n=1}^{N_d}(p(a_n|\text{expert}) + p(a_n|\text{adversary}))$$

This raises a new problem, what is the probability of an action given the expert or adversary?

1. $p(a_n|\text{adversary}) = 1$

How do we model the adversary?

$$L(\lambda|\mathcal{D}) = \prod_{m=1}^{M}\prod_{n=1}^{N_d}(p(a_n|\text{expert}) + p(a_n|\text{adversary}))$$

This raises a new problem, what is the probability of an action given the expert or adversary?

1. $p(a_n|\text{adversary}) = 1$
2. $p(a_n|\text{adversary}) = p_\psi(a_k|s_i)^{\widetilde{p}(s_i, a_k|d_m)}$

How do we model the adversary?

$$L(\lambda|\mathcal{D}) = \prod_{m=1}^{M}\prod_{n=1}^{N_d}(p(a_n|\text{expert}) + p(a_n|\text{adversary}))$$

This raises a new problem, what is the probability of an action given the expert or adversary?

1. $p(a_n|\text{adversary}) = 1$
2. $p(a_n|\text{adversary}) = p_\psi(a_k|s_i)^{\widetilde{p}(s_i,a_k|d_m)}$
3. $p(a_n|\text{adversary}) = 1 - p_\lambda(a_k|s_n)^{\widetilde{p}(s_n,a_k|d_m)}$

How do we model the adversary?

$$L(\lambda|\mathcal{D}) = \prod_{m=1}^{M}\prod_{n=1}^{N_d}(p(a_n|\text{expert}) + p(a_n|\text{adversary}))$$

This raises a new problem, what is the probability of an action given the expert or adversary?

1. $p(a_n|\text{adversary}) = 1$
2. $p(a_n|\text{adversary}) = p_\psi(a_k|s_i)^{\widetilde{p}(s_i,a_k|d_m)}$
3. $p(a_n|\text{adversary}) = 1 - p_\lambda(a_k|s_n)^{\widetilde{p}(s_n,a_k|d_m)}$

$$L(\lambda|\mathcal{D}) = \prod_{m=1}^{M}\prod_{n=1}^{N_d}\prod_{k=1}^{A}\left(p_\lambda(a_k|s_n)^{\widetilde{p}(s_n,a_k|d_m)}\right) + \left(1 - p_\lambda(a_k|s_n)^{\widetilde{p}(s_n,a_k|d_m)}\right)$$

Latent Variable

- Need a latent variable Z to represent expert or adversary
- $Z = 1 \rightarrow \Lambda \rightarrow$ Expert
- $Z = 0 \rightarrow \Psi \rightarrow$ Adversary
- Prior distribution $\sim p(Z)$

Formulation and non-convexity

$$L(\lambda, Z|\mathcal{D}) = \prod_{m=1}^{M} \prod_{n=1}^{N_d} \prod_{k=1}^{A} \sum_{z \in Z} p(z) p_z(a_k|s_n)^{\widetilde{p}(s_n, a_k|d_m)}$$

$$\ell(\lambda, Z|\mathcal{D}) = \sum_{m=1}^{M} \sum_{n=1}^{N_d} \sum_{k=1}^{A} \log \left(\sum_{z \in Z} p(z) p_z(a_k|s_n)^{\widetilde{p}(s_n, a_k|d_m)} \right)$$

Expectation Maximization

"An elegant and powerful method for finding maximum likelihood solutions for models with latent variables is called the expectation-maximization algorithm, or EM algorithm" [1]

EM in general

The EM algorithm is as follows:

1. Initialize the parameters of our model λ^{old}
2. Evaluate $p(Z|\mathcal{D}, \lambda^{\text{old}})$
3. Evaluate $\lambda^{\text{new}} = \text{argmax}_{\lambda} Q(\lambda, \lambda^{\text{old}})$ where $Q(\lambda, \lambda^{\text{old}}) = \sum_{Z} p(z|\mathcal{D}, \lambda^{\text{old}})\ell(\lambda|\mathcal{D}, z)$
4. Check if parameters have converged, if not $\lambda^{\text{old}} \leftarrow \lambda^{\text{new}}$ and return to step 2

What does EM actually do?

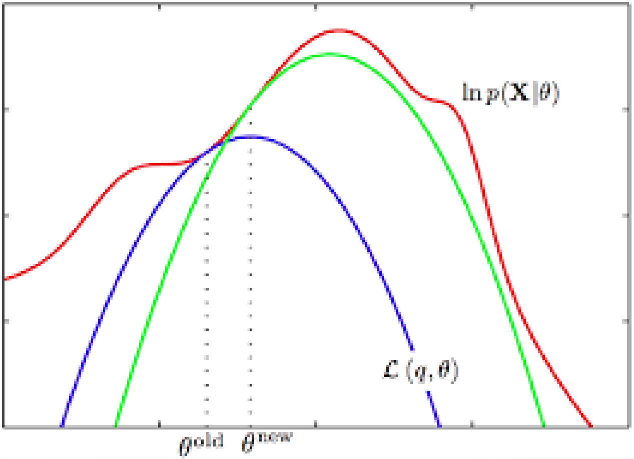


Figure: Visual Representation of EM

[6]

Expectation Step

1. Initialize λ^{old}
2. Evaluate $p(Z|\mathcal{D}, \lambda^{\text{old}})$:

$$p(Z|\mathcal{D}, \lambda) = \frac{p(z)p(d_m|z)}{\sum_{z' \in \{\lambda, \psi\}} p(z')p(d_m|z')} \quad \text{for} \quad z \in \{\Psi, \Lambda\}$$

$$p(d_m|\Lambda) = \prod_{n \in d_m} p_\lambda(a_n|s_n)$$

$$p(d_m|\Psi) = \prod_{n \in d_m} (1 - p_\lambda(a_n|s_n))$$

3. $p(\Lambda)$ represents the prior distribution over the latent variable.
   It is set by the modeler to capture how much of the Expert data is belived to be correct.
   $p(\Psi) = 1 - p(\Lambda)$

Maximization Step

1. Use $p(Z|\mathcal{D},\lambda)$ found in E-Step to computer $Q(\lambda,\lambda^{\text{old}})$

$$Q(\lambda,\lambda^{\text{old}}) = \sum_{Z} p(z|\mathcal{D},\lambda^{\text{old}})\ell(\lambda|\mathcal{D},z)$$

$$Q(\lambda,\lambda^{\text{old}}) = \sum_{m=1}^{D}\sum_{i=1}^{N_d}\sum_{k=1}^{A}\sum_{z\in Z}\left(\widetilde{p}(s_n,a_k|d_m)\log(p_z(a_k|s_n))\right)p(Z|\mathcal{D},\lambda^{\text{old}})$$

2. The above function Q is now convex and can be optiized directly $\lambda^{\text{new}} = \text{argmax}_\lambda Q(\lambda,\lambda^{\text{old}}$
3. Now set $\lambda^{\text{old}} = \lambda^{\text{new}}$
4. Repeat this until convergence.
   Convergence could either be a non increasing likelihood,
   or once all $p(Z|\mathcal{D},\lambda) \approx 1$ or 0

Results: Mountain Car

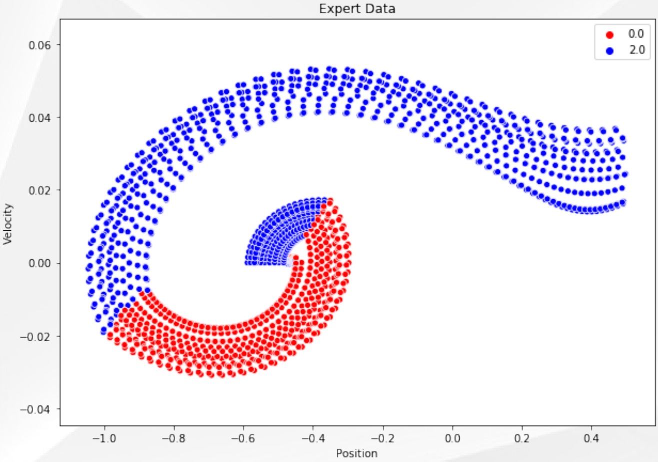
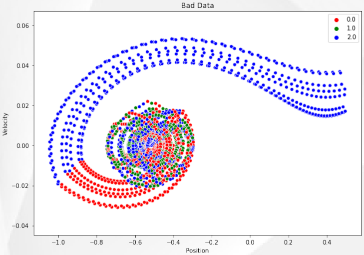
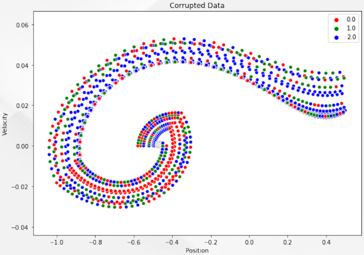


Figure: Expert Data

Results: Mountain Car



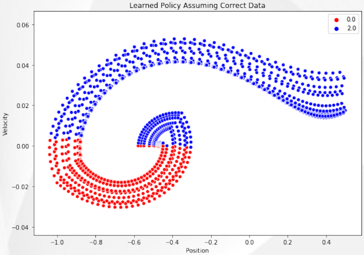
(a) label 1



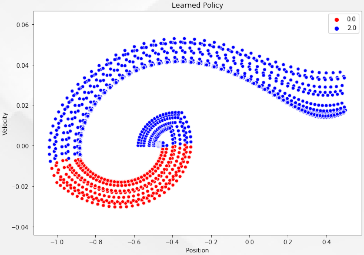
(b) label 2

Figure: Left: Half Random Policy; Right: Half Corrupted

Results: Mountain Car



(a) label 1



(b) label 2

Figure: Left: Logistic Regression; Right: R-PIL

Results: Mountain Car

Table: Average Reward Mountain Car

Adversary Type	Random Policy	Corrupted Data
Expert	-103.9 $\pm$ 6.04	-103.9 $\pm$ 6.04
R-PIL	-106.25 $\pm$ 6.00	-104.45 $\pm$ 8.50
LogReg	-119.5 $\pm$ 1.75	-132.0 $\pm$ 14.36

Results: Mountain Car

```
Expert weight for demonstration 0: [1.]
 Expert weight for demonstration 1: [1.]
 Expert weight for demonstration 2: [1.]
 Expert weight for demonstration 3: [1.]
 Expert weight for demonstration 4: [1.]
 Expert weight for demonstration 5: [1.]
 Expert weight for demonstration 6: [1.]
 Expert weight for demonstration 7: [1.]
 Expert weight for demonstration 8: [1.]
 Expert weight for demonstration 9: [1.]
 Expert weight for demonstration 10: [0.]
 Expert weight for demonstration 11: [0.]
 Expert weight for demonstration 12: [0.]
 Expert weight for demonstration 13: [0.]
 Expert weight for demonstration 14: [0.]
 Expert weight for demonstration 15: [0.]
 Expert weight for demonstration 16: [0.]
 Expert weight for demonstration 17: [0.]
 Expert weight for demonstration 18: [0.]
 Expert weight for demonstration 19: [0.]
```

(a) label 1

```
Expert weight for demonstration 0: [1.]
 Expert weight for demonstration 1: [1.]
 Expert weight for demonstration 2: [1.]
 Expert weight for demonstration 3: [1.]
 Expert weight for demonstration 4: [1.]
 Expert weight for demonstration 5: [1.]
 Expert weight for demonstration 6: [1.]
 Expert weight for demonstration 7: [1.]
 Expert weight for demonstration 8: [1.]
 Expert weight for demonstration 9: [1.]
 Expert weight for demonstration 10: [0.]
 Expert weight for demonstration 11: [0.]
 Expert weight for demonstration 12: [0.]
 Expert weight for demonstration 13: [0.]
 Expert weight for demonstration 14: [0.]
 Expert weight for demonstration 15: [0.]
 Expert weight for demonstration 16: [0.]
 Expert weight for demonstration 17: [0.]
 Expert weight for demonstration 18: [0.]
 Expert weight for demonstration 19: [0.]
```

(b) label 2

Figure: Left: Corrupted Weights; Right: Random Weights

Results: Lunar Lander

Table: Average Reward Lunar Lander

Adversary Type	Random Policy	Corrupted Data
Expert	254.45 ± 57.09	254.45 ± 57.09
R-PIL	226.49 ± 49.52	195.06 ± 64.45
LogReg	-342.64 ± 35.53	-22.24 ± 48.51

Results: Acorbot

Table: Average Reward Acrobot

Adversary Type	Random Policy	Corrupted Data
Expert	-80.65 ± 3.49	-80.65 ± 3.49
R-PIL	-92.85 ± 23.81	-102.7 ± 22.62
LogReg	-397.65 ± 85.18	-102.05 ± 31.67

Summary

- Probabilistic framework that is robust to adversarial demonstrations
- Sample and time efficient
- Could be generalized to other frameworks

Citations

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[2] Jonathan Ho and Stefano Ermon. *Generative Adversarial Imitation Learning*. 2016. arXiv: `1606.03476 [cs.LG]`.

[3] Ahmed Hussein et al. "Imitation Learning: A Survey of Learning Methods". In: *ACM Comput. Surv.* 50.2 (Apr. 2017). ISSN: 0360-0300. DOI: 10.1145/3054912. URL: https://doi.org/10.1145/3054912.

[4] Mostafa Hussein et al. *Robust Maximum Entropy Behavior Cloning*. Dec. 2020. URL: http://www.robot-learning.ml/2020/files/D1.pdf.

[5] Dean A Pomerleau. "Efficient training of artificial neural networks for autonomous navigation". In: *Neural computation* 3.1 (1991), pp. 88–97.

[6] David Rosenberg. *Expectation Maximization Algorithm*. URL: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTusXjop0MwIn0cJNCeN0n8P0a7GyWYfVeufg&usqp=CAU.