Pruning Inconsistent Demonstrations for Augmented Imitation Learning

Tu Trinh
tutrinh@berkeley.edu
5th Year EECS M.S.
University of California, Berkeley

Samarth Goel
sgoel9@berkeley.edu
5th Year EECS M.S.
University of California, Berkeley

Jonathan Guo
jonguo6@berkeley.edu
EECS B.S.
University of California, Berkeley

Abstract

Imitation learning serves as a baseline for expert-based reinforcement learning techniques and is at the forefront of several cutting-edge research problems. Real-world uses of imitation learning, however, can suffer from a lack of high-quality expert data and even from suboptimal expert data, where experts may not make correct decisions or be in disagreement with one another. How to best handle these suboptimal demonstrations is an active, open area of research; in this report, we present the first (to the best of our knowledge) approach that prunes suboptimal demonstrations from the training set first before starting the imitation learning process.

We introduce *Pruning Inconsistent Demonstrations for Augmented Imitation Learning* (PIDAIL), a technique meant to distill an expert dataset into its most useful components. PIDAIL operates under the assumption that while some demonstrators may understand at a high level what an ideal policy is for an environment, they can still be inconsistent when actually giving demonstrations, either because they are indifferent about which actions to take in certain states or because they are uncertain or indecisive about which specific action is the best to take, both leading to potentially suboptimal demonstrations. With PIDAIL, we use heuristics such as dispersion (variance or entropy) or frequency to identify which actions are most likely to suboptimal and prune the corresponding demonstrations from the agent's training set. We accomplish this through a variety of observation-clustering algorithms (agglomerative and k-means) and pruning algorithms based on cluster dispersion, outcome dispersion, action frequency, and action value.

We tested our pruning methods on several different environments from OpenAI Gym, ensuring both discrete and continuous action spaces, using four different downstream imitation learning-based algorithms: behavior cloning (BC), deep Q-learning from demonstrations (DQfD), generative adversarial imitation learning (GAIL), and soft Q imitation learning (SQIL). Our results indicate that PIDAIL can enable faster learning and higher returns for almost all environments. Overall, since PIDAIL allows the agent to separate good demonstrations from bad demonstrations prior to training without needing additional environmental interactions, it can save time, effort, cost, and computational resources.

1 Introduction

Reinforcement learning (Sutton & Barto (1998)) has provided the community with many avenues through which an AI agent or robot can learn how to navigate an unknown environment from scratch. However, doing so can be very costly, both in time and compute resources, and dangerous, such as in environments where the mistake cost is high. One alternative to pure reinforcement learning to avoid these risks is imitation learning (IL) (Bojarski et al. (2016); Ravichandar et al. (2020); Argall et al. (2009)) where the agent is given demonstrations to follow, where a demonstration is usually in the form of a complete trajectory that a human has executed in the environment. However, the use of imitation learning to guide agent learning necessitates that the demonstrations are given by an expert and are optimal. As such, these methods can fall apart if the demonstrator is not an expert and there is no access to otherwise expert demonstrators. While prior work has studied ways to handle suboptimal demonstrations via ranked preferences or relabeling done by a knowledgeable expert, this requires additional training time and effort and may not always be feasible. In such cases, the AI agent itself must try to estimate which of its provided demonstrations are optimal enough to use for training and which are not.

In this report, we focus on the particular problem of handling demonstrator suboptimality in which the demonstrator is aware of the true reward function to an extent or at a high level but is unable to truly act consistently towards this ideal. Our belief is that under this model, demonstrations in environment states where the demonstrator is more certain of the correct way to act will have lower dispersion between the actions than demonstrations in states where the demonstrator is indecisive or indifferent. This would mean that a high dispersion between demonstrated actions for the same environment state or a group of similar environment states would likely mean the demonstrator was potentially acting suboptimally or the demonstration data is unusable, cases that the agent must handle before commencing training.

Our core contribution, then, is PIDAIL (**p**runing **i**nconsistent **d**emonstrations for **a**ugmented **i**mitation **l**earning) a pruning step that the AI agent can use to filter out these suboptimal or confusing demonstrations from the training dataset, as we believe that having fewer examples from which to learn is better than learning from the wrong examples. In this report, we show several pruning methods that can be applied in different scenarios and under different environment assumptions and show that by accounting for demonstration dispersion, we are able to outperform imitation learning-based methods with unpruned demonstrations. In other words, we show that in the face of suboptimal demonstrations, AI agents and robots can use our pruning methods to, in a way, bring the demonstration set back up to optimal. To the best of our knowledge, this is the first work to study how to filter suboptimal demonstrations from a training set before starting any imitation learning, with particular focus on demonstrator inconsistency.

2 Background

A Markov Decision Process (MDP) is 5-tuple $\langle S, A, R, T, \gamma \rangle$, where S is the set of states and A is the set of actions. $R : S \times A \times S \to \mathbb{R}$ serves as a reward function for action a taken at state s, with the resulting state s'. $T : S \times A \times S \to [0, 1] = P(s' \mid s, a)$ is the transition function: if action a is taken at state s, $T(s, a, s')$ outputs the probability that s' is the next state. $\gamma \in [0, 1]$ describes a discount factor over future rewards and is applied at every timestep. In reinforcement learning literature, an agent specifies a policy $\pi : S \to A$ which is used by the agent to decide on an action given an input state. Note that policies can be either deterministic or probabilistic. For a given policy $Q^\pi(s, a)$, values are used to refer to the expected (discounted) sum of rewards after taking action a at state s, with future actions coming from the policy π. Since the optimal policy π^* maximizes expected future rewards by definition, we can show $\pi^*(s) = \arg\max_{a \in A} Q^{\pi^*}(s, a)$. Q-values are defined recursively using the Bellman equation:

$$Q^{\pi^*}(s, a) = \mathbb{E}\left[\sum_{s' \in S} P(s' \mid s, a)\left(R(s, a, s') + \gamma \max_{a' \in A} Q^{\pi^*}(s', a')\right)\right]$$

When the MDP is discrete (S and A are finite), it is possible to store all of the Q values in a tabular format. In the case of continuous states, however, this is no longer possible. Recent work in deep reinforcement learning (DRL) has focused on using deep neural networks to approximate and learn Q functions.

When the action space is discrete, we can use a deep Q network (DQN) to learn the Q values of all of the actions given an input state. Target values are given by the right-hand side of the Bellman equation; to compute the max Q value at s', a separate target network is most commonly used. The target network is updated every τ steps with the primary network.

We now briefly overview the imitation learning-based algorithms used in our experiments. DQN can be extended with expert demonstrations to achieve the Deep Q-Learning from Demonstrations (DQfD) (Hester et al. (2017)) algorithm. DQfD utilizes a pre-training phase using expert data and utilizes a temporal difference (TD) loss with regularization terms meant to prioritize the use of expert transitions. These techniques alter the agent's task from purely maximizing future rewards and make it focus on alignment with the expert's decisions, reducing the distributional distance between the agent and expert policies.

Behavioral Cloning (BC) (Bain & Sammut (1995)) is one of the original approaches taken to apply imitation learning in a reinforcement learning setting. The BC algorithm uses a dataset $D = \{(s_1, a_1), (s_2, a_2), ..., (s_n, a_n)\}$ of expert transitions to minimize the error between the actions given by an agent's policy and the expert's actions. BC's performance is capped by the expert and is susceptible to suboptimality in the expert dataset, making the advantages of demonstration pruning for higher-quality expert data clear from the standpoint of the algorithm itself.

Generative Adversarial Imitation Learning (GAIL) (Ho & Ermon (2016)) adopts the idea of Generative Adversarial networks (GANs) (Goodfellow et al. (2014)) for use in reinforcement learning by treating the policy as a generator and training a discriminator to distinguish between expert and non-expert actions using state-action pairs. GAIL is a form of inverse reinforcement learning, where a policy is trained on expert behavior without having access to a defined reward function.

Soft Q Imitation Learning (SQIL) (Reddy et al. (2019b)) is a type of imitation learning that performs on par with GAIL. In SQIL, the rewards of transitions in the replay buffer are overwritten. Expert transitions are given a reward of +1, while the agent's own transitions during training are given a reward of 0. This encourages the agent to mimic the expert's behavior on previously seen states and incentivizes the agent to go back to previously seen states, helping to negate distributional shifts. However, the main drawback with SQIL is that it requires the expert transitions to be optimal - if the expert is suboptimal, then SQIL will also be suboptimal.

3 Related Works

Much prior work has attempted to solve the problem of imitation learning from suboptimal demonstrations, using a variety of methods. One of the most popular strategies is to modify the objective function or constraints to accommodate the suboptimal demonstrations. For example, Yu et al. (2023) and Bashiri et al. (2021) transform the loss function by either relaxing the KL-divergence constraint between the expert policy and the agent policy or making the loss function into a convex optimization problem over a collection of state and action spaces. Other methods such as Zhang et al. (2022) and Li et al. (2018) have the agent adapt to the suboptimal demonstrations at the same time as learning a policy, based on some metric of confidence or performance of the demonstrated actions. However, these methods can be quite complex to implement and deploy in the real world, especially if they require extensive manipulations of existing algorithms. They can also be more computationally intensive as they must keep track of more parameters throughout training to account for the suboptimal demonstrations.

Some prior work takes extra effort to explicitly distinguish between suboptimal and optimal demonstrations. For example, Xu et al. (2022) offers an algorithm that trains a discriminator to reweight the behavioral cloning loss according to whether or not the data at hand is optimal. Our work takes

this distinguishment further by removing the need to account for any suboptimality during training as we filter out such demonstrations in a preprocessing step.

A related problem, inverse reinforcement learning (IRL) with suboptimal demonstrations has also been studied. Many popular strategies require generating synthetic demonstrations or trajectories based on the provided data and then attempting to learn the ground-truth reward function, and subsequently its optimal policy, from the augmented dataset. For example, Chen et al. (2020) and Chen et al. (2021) bootstrap from the existing demonstrations to generate new optimal trajectories, while Brown et al. (2019) uses ranked demonstrations and extrapolates beyond them to synthesize optimal ones. However, these methods were developed specifically for IRL, and while processing of suboptimal data can be done for either IL or IRL, generating synthetic data from existing non-expert data may introduce even more noise and instability during agent learning.

Other prior work has attempted to solve the suboptimal demonstration in the IL problem by assuming there exists a specific demonstrator bias. For example, Reddy et al. (2019a) assumes the demonstrator possesses an internal world model misspecification, such that their demonstrations are optimal for what they believe the world to be, while much other work assumes the demonstrator is suboptimal according to some rationality model, Boltzmann or otherwise (Laidlaw & Dragan (2022), Jeon et al. (2020)). Our work follows this avenue as we assume the demonstrator has a specific bias, one where they are fully rational in their belief and specification but are still inconsistent in providing demonstrations that follow the optimal ideal.

4 Methodology

To prune expert demonstrations, we first group observation vectors before applying pruning algorithms to either groups at large or within these groups. We try out several strategies to test what works best for the reinforcement learning environments we study. The goal is to increase the quality of our expert dataset, in turn increasing agent training speed or agent returns.

4.1 Clustering Similar Observations

The PIDAIL algorithm is based on identifying states with high indecision or uncertainty (or expert "confusion") as captured by the variance or entropy over expert actions. To compute this confusion, we first cluster similar observations together—due to the continuous nature of the observations in our environments, it makes more sense to examine how the demonstrator acted in *groups* of similar states, rather than *individual* states, as it is possible that any unique vector will only appear once in the demonstration dataset, rendering it impossible to determine expert confusion.

4.1.1 Agglomerative Clustering

We explored using agglomerative clustering without a pre-defined number of clusters to group together our observations. We vary clustering via distance metric ($L1$ norm, $L2$ norm, cosine distance), linkage (average, max, min), and distance threshold as a cutoff for grouping. Agglomerative clustering has the advantage of not needing to specify the number of clusters before grouping (as K-means needs) but creates difficulty with hyperparameter tuning due to its varying nature as distance thresholds change depending on the environment or domain knowledge of what similarity means for observations.

4.1.2 K-Means Clustering

We also explored K-means clustering as a way to group observations, setting the number of clusters after the grouping algorithm as an environment-level hyperparameter. K-means assumes that all features in the observation space have similar variance and that their priors are equal, which applies best to environments where the observation vector corresponds to location in space rather than agent attributes such as rotation or speed.

4.2 Pruning Demonstrations

After grouping expert observations to effectively segment the observation space, we apply multiple pruning algorithms at both the demonstration level and the group level to reduce the number of adverse or uncertain transitions within our dataset. Note that while we cluster based on observation s similarity, we place entire transitions—(s, a, s', r) tuples—in a cluster to make pruning easier.

4.2.1 Group Dispersion

In this pruning method, we prune entire transition clusters based on the dispersion of actions within the cluster. Demonstrations whose cluster has an action variance or entropy above a percentile threshold X will be removed from the training dataset. Our insight here is that with high variance or entropy of demonstrated actions, the agent would not be sure which are truly optimal and guessing could be detrimental, so it is better to remove such demonstrations completely.

Data: Group g
Compute variance (or entropy) of actions of group g;
if *Measure is greater than threshold* **then**
  Group has been pruned;
end
else
  Group has passed pruning;
end

Algorithm 1: Pruning groups based on action distribution

4.2.2 Outcome Dispersion

In this pruning method, we prune entire transition clusters based on the dispersion of future transitions of the trajectory. That is, for each observation s_t in a cluster, we collect the observation s_{t+H} that is a horizon H forward in time in the same trajectory; we call this collection the outcome group. We then prune observation clusters whose corresponding outcome groups have a variance or entropy that is above some percentile X compared to the rest of the outcome groups. This pruning method can be a more refined strategy because it potentially allows the agent to retain more training data if the dispersion of actions in a cluster ultimately does not have an effect on the outcome (i.e. the corresponding outcome group has low variance or entropy). This method allows the agent to distinguish between the types of expert confusion: if the outcome group ultimately has low dispersion, it can be an indicator that the demonstrator was indifferent with regard to which action they took in the original states as any of their demonstrated actions would have achieved their intended outcome. If the outcome group ultimately has high dispersion, it can be an indicator that the demonstrator was indecisive or uncertain about what to do in the original state and just made split-second decisions or tried random actions when giving demonstrations. As a result, these demonstrations would not be quality enough to use.

Data: Array of transitions (s, a, s', r) in a group
Future states = [];
for *Transition* $t = (s, a, s', r)$ *in array* **do**
  Find t's trajectory;
  Find s_H, the transition in the trajectory that occurs H steps after s;
  Add s_H to Future states
end
Compute variance or entropy of Future states;
if *Measure is greater than threshold* **then**
  Group has been pruned;
end
else
  Group has passed pruning;
end

Algorithm 2: Pruning groups based on outcome

4.2.3 Action Q-Value

In this method, we prune transitions from *within* a group, as opposed to pruning entire groups, based on the action Q-values. We first train a small Q-value estimator (either a simple DQN or SAC model) using only the demonstrated transitions as data then prune transitions from the group if the action taken does not have a high enough Q-value. More specifically, in discrete environments, we find the highest Q-value action for the observation cluster and prune transitions whose actions do not match. In continuous environments, we find all the Q-values for the cluster's transitions and prune those whose actions are not part of the top X percentile of Q-values. This method ensures that even in the face of a wide variety of actions to learn from in the demonstration data, the agent ends up only learning from the best actions.

Data: Transitions $t = (s, a, s', r)$ in a group
Compute Q values using only the transitions (either by training a neural network or computing the discounted sum of rewards along trajectories);
for *Group* g **do**
  if *Discrete* **then**
    $a_{best} = \arg\max_a Q(s, a)$ for $s \in g$;
    Result: Transitions with action a_{best}
  end
  else
    Compute $Q(s, a)$ for all transitions in group g;
    Result: Transitions with Q value in the top X percentile
  end
end

Algorithm 3: Pruning transitions based on Q value

4.2.4 Action Frequency

In this method, we prune transitions within a group based on their associated action. For discrete action spaces, we sort the actions within the group by how many times they were taken. Then, transitions with actions in the top X percentile of actions are kept, while the rest are pruned. We found this more flexible than taking just the mode action since we can just select the mode action by taking $X = \epsilon$ for some $\epsilon > 0$.

Data: Array of transitions (s, a, s', r) in a group
Sort array based on frequency of a;
Good actions = actions in top X percentile of actions taken;
Result: Array of transitions with $a \in$ Good actions

Algorithm 4: Pruning transitions based on action - discrete case

For continuous action spaces, we first compute the mean action. Then, we compute the L_2 norm between actions and the mean action and only keep the bottom X percentile of this norm. Therefore, transitions whose actions are far away from the mean are pruned. This reduces the variance of the actions taken by this group.

Data: Array of transitions (s, a, s', r) of a group
Compute mean action $\bar{a} = \frac{1}{n}\sum_{i=1}^{n} a_i$;
for *Transition* $t = (s, a, s', r)$ *in array* **do**
  Compute $d(a, \bar{a}) = ||a - \bar{a}||$;
end
Sort array based on of $d(a, \bar{a})$;
Result: Array of transitions with $d(a, \bar{a})$ in the bottom X percentile

Algorithm 5: Pruning transitions based on action - continuous case

4.3 Performance Improvements

In this section, we theoretically analyze how our pruning approach can help the AI agent reach higher environment returns in fewer training steps. The central idea of our approach is to reduce number of the expert trajectories in a way that provides a more consistent set of actions given various observation subsets.

For example, in the action frequency pruning, within each group g, $\text{Var}(a \mid s, g') < \text{Var}(a \mid s, g)$, where $g' \subset g$ is the group after pruning. To see this, note that within each group, actions that are pruned contribute higher than average squared error. This means that the remaining actions will have lower average squared error, which means lower variance. Then, because each group is independent, we can sum over all groups to obtain

$$\text{Var}(a \mid s, D') < \text{Var}(a \mid s, D)$$

where D is our entire dataset before pruning and D' is our dataset after pruning. By reducing the variance of the action distribution in our expert data, our agents will learn faster.

5 Experiments

5.1 Experimental Design

To show the feasibility of our pruning methods on a wide variety of applications, we tested them empirically on different combinations of imitation learning-based methods and environments, using OpenAI Gym (Brockman et al. (2016)), aiming to have a balance between environments with discrete action spaces and continuous actions spaces. We tested our methods on the following environments: Cartpole, Lunar Lander, Ant, Hopper, Walker, and Half Cheetah.

For environments with discrete action spaces, we tested our pruning methods with behavioral cloning (BC) and deep Q-learning from demonstrations (DQfD). For environments with continuous action spaces, we tested our pruning methods with BC, generative adversarial imitation learning (GAIL), soft Q imitation learning (SQIL).

Because we are studying how the agent itself may handle a set of suboptimal demonstrations without needing to query anything from a human, we ensured that none of these algorithms required an

interactive expert to stand by or supervise the AI agent in any way. As such, while high-performing, methods like DAgger (Ross et al. (2010)) were not included in our study.

To emulate near-optimal and suboptimal demonstrators, we gathered such data by either (1) executing existing policies that were known to work and then adding noise to the trajectories or (2) training policies from scratch ourselves but not until full convergence (Raffin et al. (2021); Gleave et al. (2022)). Afterward, we tested each of the four algorithms both with and without prior demonstration pruning, training all of these until convergence.

5.2 Results and Analysis

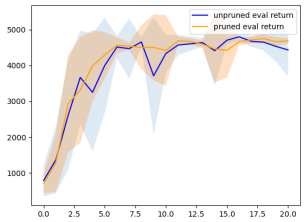

Figure 1: Ant, BC, K-means clustering, Action frequency pruning

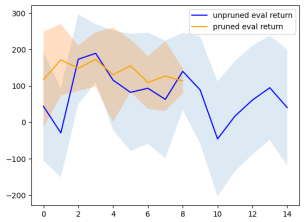

Figure 2: Lander, DQfD, K-means clustering, Outcome dispersion pruning

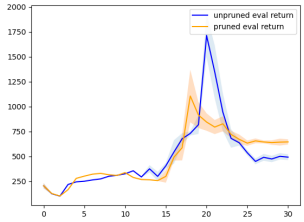

Figure 3: Hopper, GAIL, Agglomerative clustering, Group dispersion pruning

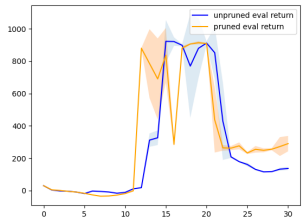

Figure 4: Walker, GAIL, Agglomerative clustering, Action value pruning

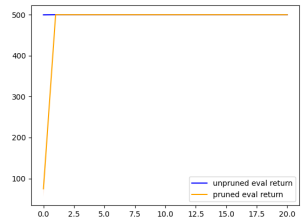

Figure 5: Cartpole, BC, Agglomerative clustering, Action value pruning

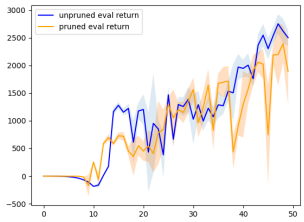

Figure 6: Cheetah, SQIL, K-Means, Outcome dispersion pruning

In Figures 1-6, we show our experiment results, each averaged over at least 20 different runs. The x axis in each plot is the number of training epochs and the y axis is the eval environment returns; the last point on each plot is how the agent performed after all training was complete. The blue lines show how the agent learned without any pruning while the orange lines show how the agent learned with clustering and pruning. Due to space constraints we only show a sample of our results, aiming to display a wide variety of algorithm, clustering, and pruning combinations, though on average most of our experiments led to similar results for the corresponding environment and imitation learning algorithm.

In most plots, the agent with pruning reaches higher performance earlier than the agent without pruning, showing that pruning helps our agents learn and train faster. One interesting aspect to note is that because our replay buffer has fewer transitions after pruning for some agents, each epoch will also take less time to train. In addition, by the end of training and often several epochs before that, the agent with pruning almost always achieves higher returns than the agent without pruning (except in cartpole which is a toy environment that is easy for any algorithm to achieve full returns). Finally, as shown by the standard errors, our methodology also helps reduce the returns variance attained during both training and evaluation. Overall, our results show there exists a promising, useful benefit to pruning inconsistent demonstrations before commencing training.

In some cases (such as HalfCheetah), where our expert dataset did not have that many suboptimal trajectories, the level of pruning would have also reduced the number of overall, "good-enough" trajectories. This would have hurt the overall performance of the agent after pruning. Indeed, we used percentiles as thresholds for all of our pruning methods as we assumed the expert demonstrations would all be noisy or suboptimal. However, as our HalfCheetah experiment showcased, when most demonstrations were (near-)optimal enough to begin with, it could be a drawback. As such, we would advise practitioners to choose different threshold settings based on their domain knowledge and demonstration quality.

6 Discussion

6.1 Limitations and Future Work

The main limitation we encountered was the lack of stable, robust compute, which limited the number and length of experiments we were able to run as we conducted experiments solely on local laptops. Nevertheless, we tried our best to run as many training steps as possible on as many environments as possible.

One area of future work could be using our pruning methods but with adaptive thresholds based on the perceived optimality of the demonstrator. Existing work shows promising results of being able to determine the demonstrator (Beliaev et al. (2022); Ghosal et al. (2023)); this process can be combined with ours to prune demonstrations in a more nuanced way. For example, for demonstrators that are perceived to be more optimal, we can perhaps afford to be more lenient with keeping demonstrations, while for demonstrators that are perceived to be less optimal, we can perhaps be more liberal with pruning. In addition, while we originally designed our pruning methods to be used in imitation learning settings to ensure the AI agent would only be learning from the most likely optimal demonstrations, these methods can be applied to other reinforcement learning settings that involve learning from demonstrations. For example, prior pruning of demonstrations before conducting inverse reinforcement learning could help the agent converge to the estimated ground-truth reward function faster and more accurately.

Our work could also be extended to methods like DAgger that require expert supervision. If no optimal experts are available, perhaps our pruning methods could be used with suboptimal experts to supervise agent training.

Lastly, we would want to explore new environments outside the toy gym environments on datasets where expert suboptimality is more of a practical concern. These could include learning from human demonstrations in a variety of settings including robots, autonomous vehicles, and even classification datasets that leverage crowdsourced human labels.

6.2 Conclusion

In this report, we propose, demonstrate, and analyze methods to prune inconsistent demonstrations from an AI agent's training dataset to augment imitation learning. We show that our methods can enable faster and better learning in a variety of environments and with a variety of downstream imitation learning-based algorithms. Future roboticists and researchers will be able to use these methods to ensure agents are always learning from quality demonstrations, no matter if there is only access to suboptimal, non-expert ones.

Group Contributions

Alina: Created project proposal and guided research and experiments direction. Implemented DQfD algorithm, some pruning methods, and common/utility functionalities. Wrote abstract, introduction, related works, methodology, experiments, and discussion sections.

Samarth: Set up agent training and evaluation scripts, set up Behavioral Cloning, GAIL, and SQIL algorithms using the stable_baselines3 and imitation libraries. Also set up clustering methods. Wrote abstract, background, and methodology sections.

Jonathan: (Joined group after milestone submission.) Gathered all expert data and developed some pruning algorithms. Wrote abstract, background, related works, methodology, and experiments sections.

References

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, 1995. URL https://api.semanticscholar.org/CorpusID:10738655.

Mohammad Ali Bashiri, Brian Ziebart, and Xinhua Zhang. Distributionally robust imitation learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 24404–24417. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/cc8090c4d2791cdd9cd2cb3c24296190-Paper.pdf.

Mark Beliaev, Andy Shih, Stefano Ermon, Dorsa Sadigh, and Ramtin Pedarsani. Imitation learning by estimating expertise of demonstrators, 2022.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016. URL http://arxiv.org/abs/1604.07316.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL http://arxiv.org/abs/1606.01540.

Daniel S. Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations, 2019.

Letian Chen, Rohan Paleja, and Matthew Gombolay. Learning from suboptimal demonstration via self-supervised reward regression, 2020.

Letian Chen, Rohan Paleja, and Matthew Gombolay. Towards sample-efficient apprenticeship learning from suboptimal demonstration, 2021.

Gaurav R. Ghosal, Matthew Zurek, Daniel S. Brown, and Anca D. Dragan. The effect of modeling human rationality level on learning rewards from multiple feedback types, 2023.

Adam Gleave, Mohammad Taufeeque, Juan Rocamonde, Erik Jenner, Steven H. Wang, Sam Toyer, Maximilian Ernestus, Nora Belrose, Scott Emmons, and Stuart Russell. imitation: Clean imitation learning implementations. arXiv:2211.11972v1 [cs.LG], 2022. URL https://arxiv.org/abs/2211.11972.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

Todd Hester, Matej Vecerík, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John P. Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732, 2017. URL http://arxiv.org/abs/1704.03732.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *CoRR*, abs/1606.03476, 2016. URL http://arxiv.org/abs/1606.03476.

Hong Jun Jeon, Smitha Milli, and Anca D. Dragan. Reward-rational (implicit) choice: A unifying formalism for reward learning, 2020.

Cassidy Laidlaw and Anca Dragan. The boltzmann policy distribution: Accounting for systematic suboptimality in human models, 2022.

Yuxiang Li, Ian Kash, and Katja Hofmann. Learning good policies from suboptimal demonstrations. In *The 14th European Workshop on Reinforcement Learning (EWRL 2018)*, October 2018. URL https://www.microsoft.com/en-us/research/publication/learning-good-policies-from-suboptimal-demonstrations/.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. 22(1), jan 2021. ISSN 1532-4435.

Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3:297–330, 2020.

Siddharth Reddy, Anca D. Dragan, and Sergey Levine. Where do you think you're going?: Inferring beliefs about dynamics from behavior, 2019a.

Siddharth Reddy, Anca D. Dragan, and Sergey Levine. SQIL: imitation learning via regularized behavioral cloning. *CoRR*, abs/1905.11108, 2019b. URL http://arxiv.org/abs/1905.11108.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. *CoRR*, abs/1011.0686, 2010. URL http://arxiv.org/abs/1011.0686.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

Haoran Xu, Xianyuan Zhan, Honglei Yin, and Huiling Qin. Discriminator-weighted offline imitation learning from suboptimal demonstrations. 07 2022. doi: 10.48550/arXiv.2207.10050.

Lantao Yu, Tianhe Yu, Jiaming Song, Willie Neiswanger, and Stefano Ermon. Offline imitation learning with suboptimal demonstrations via relaxed distribution matching, 2023.

Songyuan Zhang, Zhangjie Cao, Dorsa Sadigh, and Yanan Sui. Confidence-aware imitation learning from demonstrations with varying optimality, 2022.