BAgger: A Bayesian Algorithm for Safe and Query-efficient Imitation Learning

Constantin Cronrath[1], Emilio Jorge[2,3], John Moberg[2,3], Mats Jirstrand[2,3] and Bengt Lennartson[1]

***Abstract*— Safety and query efficiency may present a challenge when learning a robot control policy with Dataset Aggregation (DAgger). We propose BAgger, an Imitation Learning algorithm that, using a Bayesian approach, aims to mitigate those challenges by predicting state novelty and policy error. In BAgger, the expert is queried only when there is a significant risk of not being able to imitate the expert, e.g. in novel parts of the state space. We present empirical results indicating that BAgger is, both, safer than DAgger and SafeDAgger on a robot control task, while still being query-efficient.**

I. Introduction

Imitation Learning (IL) has been proven an effective way to learn control policies in robotics. For instance, IL has been applied to autonomous driving [1]–[4], robotic grasping in cluttered environments [5], and inserting surgical needles [6]. The goal of IL is to train a novice controller to imitate the behaviour of an expert, which, for example, may be a human or an expensive algorithm. One way of doing this is through the Dataset Aggregation (DAgger) algorithm [7].

DAgger trains a novice controller online by applying a supervised learning algorithm to an aggregated set of state observations, each of which is labelled with the expert's control action. The expert's control of the task is annealed and the novice is given successively more control. The expert then labels all observations with the optimal control action in retrospect, which is used to update the learner. Although DAgger is popular as a benchmark in IL literature, because of its simplicity, in practice it has two limitations. First, performance, and hence safety, is only guaranteed asymptotically, and not during training. Second, each state observation must be labelled by the expert. This may be prohibitively expensive, e.g. in the case of a human expert.

These limitations, safety and query efficiency, have been addressed previously in literature such as [2], [6], [8]. Section IV provides more details on related work. We propose Bayesian dataset Aggregation (BAgger), an extension of DAgger that aims to address the aforementioned problems of safety and query-efficiency. BAgger trains a Gaussian Process (or, alternatively, a Bayesian Neural Network (BNN)) to predict the expected error between the novice and expert policies for any given state. The expert policy is only queried in states where there is a significant risk of not being able to safely imitate the expert. As illustrated in Fig. 1, our results indicate a better safety and query efficiency. BAgger achieves consistently expert-like performance – while requiring less than 40% of the queries required by DAgger.

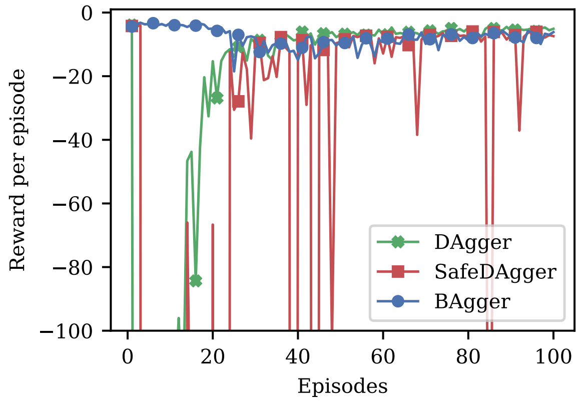

Fig. 1. Results on the OpenAI *Reacher-v2* environment, averaged over 15 evaluations. Our algorithm, BAgger, is compared to two baseline methods. Note that BAgger's performance is consistently high throughout training.

Section II introduces the notation used throughout this paper. Section III presents the BAgger algorithm, followed by a brief discussion on related work in Section IV. Empirical results of BAgger are documented in Section V, while Section VI provides some concluding remarks.

II. Preliminaries

We consider a sequential decision making problem with Markovian dynamics. Following the notation of [9], the learner encounters states $s \in \mathcal{S}$, takes actions $a \in \mathcal{A}$, and thereby influences the distribution of next states s'. A policy $\pi \in \Pi$ is a mapping of states to actions ($\pi : \mathcal{S} \to \mathcal{A}$). We denote the policy of the expert with π^*, and the policy of the novice with $\hat{\pi}$. A roll-out is the application of a policy π to sample a T-step trajectory of state-action pairs $\langle s, a \rangle$.

III. Algorithm

DAgger randomly blends the expert and novice policies during training. For that reason, the novice might be given control in states that are *unsafe*. Following [6], a state is considered *unsafe*, if it is novel (unseen) to the novice, or if $\hat{\pi}_i$ deviates too far from π^*. We formally define the deviation of the two policies as

$$\epsilon(s) := \|\hat{\pi}_i(s) - \pi^*(s)\| \quad \forall s \in \mathcal{S}$$

*This work was financed by The Swedish Foundation for Strategic Research, through the Smart Assembly 4.0 project, within the Winquist Laboratory and Fraunhofer Cluster of Excellence *Cognitive Internet Technologies*. The support is gratefully acknowledged.

[1]Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden {cronrath, bengt.lennartson}@chalmers.se
[2]Fraunhofer-Chalmers Centre for Industrial Mathematics, Gothenburg, Sweden {emilio.jorge, john.moberg, mats.jirstrand}@fcc.chalmers.se
[3]Fraunhofer Center for Machine Learning

where $\|\cdot\|$ may be any error metric, e.g. the L_2-norm. To obtain an estimate of the expected error also in novel states without querying the expert, we fit a function $\hat{\psi}(s)$ to the error. $\hat{\psi}(s)$ will help us decide whether we can trust the novice or need to query the expert during roll-out. In low-risk states, the novice can be given control, and the expert must not demonstrate the optimal action. Therefore, it is crucial for safety to be able to assess the confidence in the expected error prediction – especially when in novel states. Regular Neural Networks (NNs) tend to be overly confident when extrapolating to novel states and only provide an estimate of the mean [10]. We therefore utilise a Gaussian Process [11], or respectively its approximation as a Bayesian Neural Network (BNN), for the implementation of $\hat{\psi}$.

Gaussian Processes (GP) take a Bayesian approach to supervised learning. Instead of providing only the mean of a prediction, a GP also gives a normal distribution of possible values. The variance of this distribution can be interpreted as the confidence in a prediction and depends on the evidence observed in that region. Predictions in known state regions, with plenty of evidence, are expected to exhibit lower variance than predictions far into novel state regions. The variance in the error prediction can, thus, also serve as a proxy measure for the novelty of a state.

The mean $\hat{\mu}_\epsilon(s)$ and the variance $\hat{\sigma}^2_\epsilon(s)$ of the error in some state s are estimated by the corresponding sample statistics applied to a sample from $\hat{\psi}(s)$ of size M. With these, we define the BAgger policy:

$$\pi(s) = \begin{cases} \hat{\pi}_i(s) & \text{if } \hat{\mu}_\epsilon(s) + 2\hat{\sigma}_\epsilon(s) \leq \tau, \\ \pi^*(s) & \text{otherwise.} \end{cases}$$

Assuming a normally distributed predicted error[1], an upper confidence bound of the confidence interval $\hat{\mu}_\epsilon(s) + 2\hat{\sigma}_\epsilon(s)$ covers over 97% of the confidence interval. An UCB below a given safety threshold τ hence indicates a *safe* state, in which the expert must not be queried to improve sample efficiency. Vice versa, a UCB above τ implies an *unsafe* state, either due to a large mean error (novice and expert policy deviate too much in that state), or due to a high variance (the state may be novel). In such states, the expert is queried for the optimal, safe action. Therefore, the safety threshold parameter τ needs to be chosen to strike a good balance between safety and query efficiency.

Having defined the BAgger policy, the full BAgger algorithm is presented in Algorithm 1. Additions to the original DAgger algorithm are highlighted in blue for clarity.

IV. Related Work

Safety and query efficiency have been addressed in prior work. Chernova and Velosa fit Gaussian Mixture Models (GMM) over the actions [12]. Each state is classified as belonging to one particular action. If the confidence, given by the GMM, is below a given threshold, the expert is queried for the optimal action. Unlike BAgger, this approach is limited to discrete action spaces.

[1]While the error is strictly positive this is still an acceptable approximation since the important cases are where the error is large.

Algorithm 1 BAgger

```
Require: $\tau$
  Initialise $D \leftarrow \emptyset$
  Initialise $\hat{\pi}_1$ to any policy in $\Pi$
  Initialise $\hat{\psi}_1$
  for $i = 1$ to $N$ do
    Let $\pi$ be the BAgger policy
    Sample $T$-step trajectory using $\pi$
    Get $D_i' \leftarrow \{\langle s, a\rangle | \hat{\mu}_\epsilon(s) + 2\hat{\sigma}_\epsilon(s) > \tau \wedge a \leftarrow \pi^*(s)\}$
    Aggregate datasets: $D \leftarrow D \cup D_i'$
    Train $\hat{\pi}_{i+1}$ on $D$
    Train $\hat{\psi}_{i+1}$ on $\{\langle s, \epsilon(s)\rangle | s \in D\}$
  end for
  return best $\hat{\pi}_i$ on validation.
```

Query efficiency also motivates Disturbances for Augmenting Robot Trajectories (DART) [13]. It is argued that in DAgger, the novice may wander off into irrelevant state regions. Therefore, the expert's actions are perturbed with noise to force the expert to demonstrate correcting behaviour around the optimal trajectory. The novice policy is only used to estimate the noise parameter, but not for control in the environment. Similarly reasoned is Maximum Mean Discrepancy Imitation Learning (MMD IL) [1]. MMD IL trains multiple, local policies for sub-regions of the state space. In each state, the MMD-closest policy is selected from all previously trained policies. If that policy is far from the current state, the expert is queried and it is decided whether to go with expert or novice policy. However, the novice remains in control in unsafe, but known states.

Lee et al present an offline IL algorithm for model predictive control (MPC) in autonomous driving. The algorithm is paired with Reinforcement Learning to learn the uncertainty threshold for switching from novice to expert controller [3]. The states visited during roll-out are not taken into account when training the novice. As pointed out by [3], the algorithm is strictly speaking not DAgger-like. An MPC expert for autonomous driving is also used by Sampled-DAgger [14]. After the first DAgger iteration, control is given to the novice. The expert generates in parallel a trajectory with lower sample frequency. If the Euclidean distance between the trajectory generated by the novice and the one of the expert becomes too large, the expert labels the observed states and expands the novice's training set. Control is not returned to the expert in cases of large deviation.

Closest to our work are the DAgger extensions SHIV [6] and SafeDAgger [2]. SHIV (SVM-based reduction in Human InterVention) trains two one-class SVMs as decision policies. One classifies novel states, while the other classifies states that have been misclassified in the previous iteration. Taken together, the SVMs decide on the safety of a state. As the name suggests, this approach is limited by the capabilities of SVMs. Similar to BAgger, SafeDAgger [2] trains, in addition to the novice policy, a binary classifier as a safety policy. A state is classified unsafe if the error between expert and novice policy for a state is above a selected threshold. The expert is then only queried for states that the safety policy classified as unsafe. Unlike our approach, SafeDAgger performs the classification prior to training the safety policy. No explicit assumptions are made about the safety of novel states. Although the *softmax*-classification provides predictive probabilities, they can not be interpreted as model confidence [10]. Instead, SafeDAgger relies on the availability of a pre-trained, good safety policy. Pre-training in SafeDAgger requires an additional dataset D_{safe}, collected using π^*. This essentially trades off query-efficiency for safety in SafeDAgger. A variation of SafeDAgger is DropoutDAgger [8]. The novice policy is here implemented as a BNN. In states of high uncertainty in the novice policy, control is given to the expert. Query efficiency is not accounted for – the expert is queried for every state.

V. Experimental Results

We present empirical evaluations of BAgger on two environments of the OpenAI Gym [15] built on top of MuJoCo [16]. For comparison, we report the performance of DAgger [7] and SafeDAgger [2] in the same environment.

A. *Reacher-v2*

The goal in *Reacher* is to control an articulated robot arm with two degrees of freedom in a 2-dimensional plane. For each episode, a new goal location is given, which is to be reached by the tool centre point. The state space is continuous and $\mathcal{S} \subset \mathbb{R}^{11}$. The action space contains the torques of the two joints, hence $\mathcal{A} \subset \mathbb{R}^2$.

The GP in BAgger is approximated by a BNN implemented as a NN with a dropout rate of 0.2, motivated by [10]. A sample size $M = 32$ was used for the sample mean and variance. An informal hyperparameter search found $\tau = 0.15$ to strike a good balance between safety and query efficiency. Both the novice policy and the error predictor were represented by NNs with 2 hidden layers consisting of 16 ReLU units each. We used L^2 regularization with $\lambda = 0.01$ and Adam [17] for optimization.

Fig. 1 shows the reward per episode for DAgger, SafeDAgger, and BAgger during the course of 100 episodes. While all algorithms eventually learn how to solve the task, only BAgger manages a consistently high reward. In a safety-critical application, it is clear that this is a non-negotiable quality. Fig. 2 shows how the ratio of expert queries, averaged over 15 evaluations, changes over time. We see that BAgger initially makes extensive use of the expert, but the expert ratio eventually approaches 0 as the novice learns how to perform the task safely. Note that while DAgger has a decreasing expert ratio, it still queries the expert for each step to label the data.

In Table I we can compare the amount of queries to the expert for each algorithm. Both SafeDAgger and BAgger have a significantly lower amount of queries compared with DAgger. While SafeDAgger has a slightly lower query footprint compared to BAgger it is not enough to motivate the difference in safety observed in Figure 1.

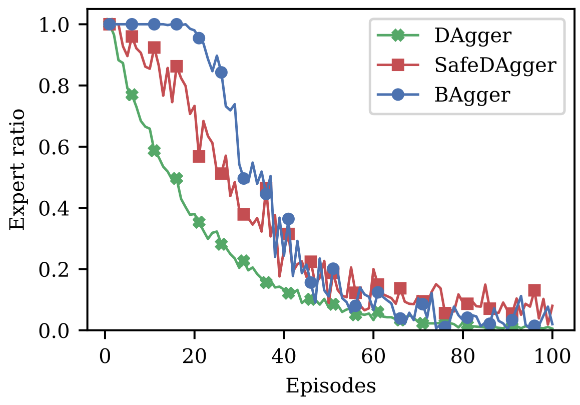

Fig. 2. Average ratio of control by the expert policy during roll-out in *Reacher-v2*. All algorithms eventually give full control to the novice.

TABLE I
AVERAGE QUERY EFFICIENCY AND CUMULATIVE REWARD

Environment	Algorithm	Avg. Queries	Avg. Cum. Reward
Reacher-v2	DAgger	5000	-14.8k
	SafeDAgger	1656	-46.7k
	BAgger	1872	-0.8k
Hopper-v2	DAgger	25355	98.4k
	SafeDAgger	9980	100.2k
	BAgger	8293	110.9k

B. *Hopper-v2*

The goal of *Hopper* is to control a simplified figure to hop forward. The state and action spaces are $\mathcal{S} \subset \mathbb{R}^{11}$ and $\mathcal{A} \subset \mathbb{R}^3$. We choose a NN with 2 hidden layers of 32 units with ReLU activation for the novice policy. For BAgger, we set $\tau = 0.5$ and train a GP for $\hat{\psi}$, using the implementation of [18]. All other settings are shared with Subsection V-A.

Table I contains averaged results over 5 evaluations of 30 episodes each. Again, we observe a significant improvement in query efficiency for SafeDAgger and BAgger compared to DAgger. The increase of average cumulative reward, as a proxy for safety, however, is less stark than in *Reacher-v2*. Whereas in *Reacher*, the design of the task requires visiting novel state regions frequently, a good policy in *Hopper* periodically visits similar states. This facilitates IL without a strong novelty detection.

VI. Concluding Remarks

Our results indicate that estimating state safety and novelty, through a Gaussian Process that learns the policy error, makes BAgger safer and more query-efficient than other IL algorithms. Especially in tasks like *Reacher*, in which visiting novel state regions is frequently required, the confidence estimates of the GP prove to be valuable for safety. How prior knowledge about the properties of the error function ϵ can facilitate setting τ and improve query efficiency further, will be addressed in future work.

References

[1] B. Kim and J. Pineau, "Maximum Mean Discrepancy Imitation Learning," *Workshop on Robot Learning, ICML*, 2013. [Online]. Available: http://roboticsproceedings.org/rss09/p38.pdf

[2] J. Zhang and K. Cho, "Query-Efficient Imitation Learning for End-to-End Autonomous Driving," in *Proceedings of the 31st AAAI Conference on Arificial Intelligence (AAAI-17)*, 2 2017, pp. 2891 – 2897.

[3] K. Lee, K. Saigol, and E. Theodorou, "Safe end-to-end imitation learning for model predictive control," in *Proceedings of Robotics: Science and Systems (RSS)*, 2018. [Online]. Available: http://arxiv.org/abs/1803.10231

[4] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, "Agile Off-Road Autonomous Driving Using End-to-End Deep Imitation Learning," *CoRR*, 9 2017. [Online]. Available: http://arxiv.org/abs/1709.07174

[5] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Learning Generalizable Robot Skills from Demonstrations in Cluttered Environments," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 10 2018. [Online]. Available: http://arxiv.org/abs/1808.00349

[6] M. Laskey, S. Staszak, W. Yu, S. Hsieh, J. Mahler, F. T. Pokorny, A. D. Dragan, and K. Goldberg, "SHIV: Reducing Supervisor Burden in DAgger using Support Vectors for Efficient Learning from Demonstrations in High Dimensional State Spaces," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5 2016, pp. 462–469. [Online]. Available: http://ieeexplore.ieee.org/document/7487167/

[7] S. Ross, G. J. Gordon, and J. A. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS) 2011*, vol. 15, 2011, pp. 627–635. [Online]. Available: http://arxiv.org/abs/1011.0686

[8] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer, "DropoutDAgger: A Bayesian Approach to Safe Imitation Learning," 9 2017. [Online]. Available: http://arxiv.org/abs/1709.06166

[9] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. MIT Press, 1998.

[10] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*, 6 2016, pp. 1050–1059. [Online]. Available: http://proceedings.mlr.press/v48/gal16.html

[11] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. MIT Press, 2006.

[12] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using Gaussian mixture models," in *Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2007.

[13] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "DART: Noise Injection for Robust Imitation Learning," in *1st Conference on Robot Learning (CoRL 2017)*, 3 2017. [Online]. Available: http://arxiv.org/abs/1703.09327

[14] L. Sun, C. Peng, W. Zhan, and M. Tomizuka, "A Fast Integrated Planning and Control Framework for Autonomous Driving via Imitation Learning," *CoRR*, 7 2017. [Online]. Available: http://arxiv.org/abs/1707.02515

[15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," 2016.

[16] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5026–5033.

[17] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 12 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in {P}ython," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.