EnsembleDAgger: A Bayesian Approach to Safe Imitation Learning

Kunal Menda,[1] Katherine Driggs-Campbell,[2] and Mykel J. Kochenderfer[1]

Abstract— **Although imitation learning is often used in robotics, the approach frequently suffers from data mismatch and compounding errors. DAgger is an iterative algorithm that addresses these issues by aggregating training data from both the expert and novice policies, but does not consider the impact of safety. We present a probabilistic extension to DAgger, which attempts to quantify the confidence of the novice policy as a proxy for safety. Our method, EnsembleDAgger, approximates a Gaussian Process using an ensemble of neural networks. Using the variance as a measure of confidence, we compute a decision rule that captures how much we doubt the novice, thus determining when it is safe to allow the novice to act. With this approach, we aim to maximize the novice's share of actions, while constraining the probability of failure. We demonstrate improved safety and learning performance compared to other DAgger variants and classic imitation learning on an inverted pendulum and in the MuJoCo HalfCheetah environment.**

I. Introduction

To be truly intelligent, robotic systems must have the ability to learn by exploring their environment and state space in a safe way [1]. One method to guide exploration is to learn from expert demonstrations [2], [3], [4]. In contrast to reinforcement learning, where an explicit reward function must be defined, imitation learning guides exploration through expert supervision, allowing a robot to effectively learn from direct experience [5]. However, such supervised approaches are often suboptimal or fail when the policy that is being trained (referred to as the novice policy) encounters situations that are not adequately represented in the dataset provided by the expert [6], [7]. While failures may be insignificant in simulation, safe learning is important when acting in the real world [1].

There are several methods for guided policy search in imitation learning settings [8]. One example is DAgger, which improves the training dataset by aggregating new data from both the expert and novice policies [7]. DAgger has many desirable properties, including online functionality and theoretical guarantees. This approach, however, does not guarantee safety. Recent work extended DAgger to address some inherent drawbacks [9], [10]. In particular, SafeDAgger augments DAgger with a decision rule policy to provide safe exploration while minimizing queries to the expert [11].

The shared goal of these methods is to efficiently train the novice to control the system while minimizing expert intervention. These algorithms assume that by allowing the novice to act, the system will likely deviate from the expert trajectory set and sample a new state. There is a chance, however, that the state visited is unsafe, or is a failure state. If the expert acts instead, we assume that the system will move along a safe trajectory, which is likely through states similar to those previously observed. The goal of this paper is to present an algorithm that maximizes the novice's share of actions, while constraining the probability of failure.

Ideally, the proximity to a failure state (measured as an l_2-distance or likelihood of encountering the state under some operating condition) is known, and a safety envelope can be computed to guarantee safety [12]. In the case of model-free learning, such guarantees are much more difficult to make. If we consider the novice action to be a perturbed form of the expert action, then we hypothesize that for many systems, the magnitude of permissible perturbation to expert actions is related to the distance from unsafe regions. Further, in a model-free case where expert demonstrations are available, we hypothesize that there is an inverse relationship between a state's similarity to those in expert trajectories and allowed perturbations. We visualize this intuition in Figure 1. In the left panel, we see that the maximum permissible deviation from an expert action should be low as the system approaches a wall, which is considered a dangerous state. In such settings, experts will likely prefer trajectories that maintain some margin of distance from unsafe states. Assuming this to be the case, it follows that in unfamiliar states, the system is likely at higher risk of entering failure states, and thus it is safer to allow the expert to act. While in familiar regions, it is permissible for the novice to act with large deviation from expert action.

This paper extends DAgger to a probabilistic domain, and aims to minimize expert intevention while constraining the likelihood of failure. While SafeDAgger uses the *discrepancy* between the expert and the novice to determine safety, we measure *doubt* by quantifying the uncertainty or confidence of the novice policy. To quantify doubt, we use an ensemble of neural networks to estimate the variance of the novice action in a particular state, which we show can effectively approximate Gaussian Processes (GPs), even in complex, high-dimensional spaces [13].

We demonstrate how our method out-performs existing DAgger variants in an imitation learning setting. This paper makes two key contributions: (1) we present EnsembleDAgger, a Bayesian extension to DAgger, which introduces a probabilistic notion of safety to minimize expert intervention while constraining the probability of failure; and (2) we demonstrate the utility of this approach with improved performance and safety in an imitation learning case study on an inverted pendulum and demonstrate the scalability of the approach on the MuJoCo HalfCheetah domain.

[1]Kunal Menda and Mykel J. Kochenderfer are at Stanford University, Stanford, CA 94305, USA {kmenda,mykel}@stanford.edu

[2]Katherine Driggs-Campbell is at the University of Illinois at Urbana-Champaign, IL 61820, USA krdc@illinois.edu

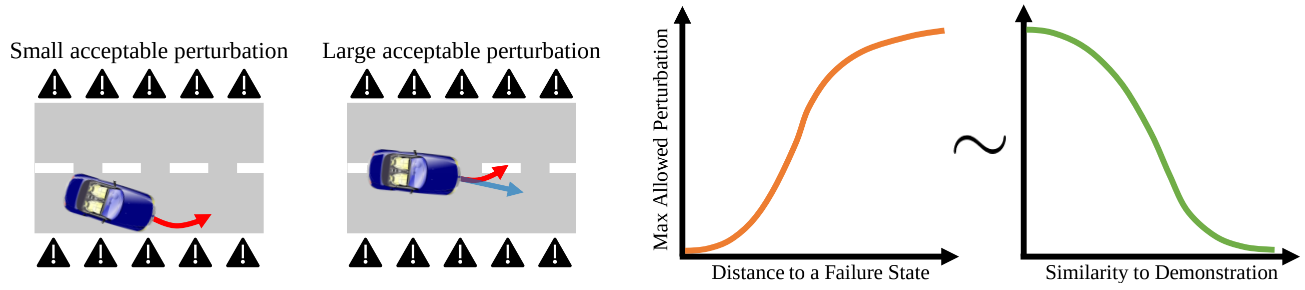

Small acceptable perturbation
Large acceptable perturbation
Max Allowed Perturbation
Distance to a Failure State
$\sim$
Similarity to Demonstration

Fig. 1: Visualization of the tradeoffs between familiarity and risk. (left) Example scenarios of where perturbations are (not) permissible due to low (high) risk. Red trajectories illustrate expert corrections and the blue trajectory illustrate novice actions. (right) Plots visualizing the ideal tradeoff between distance to failure state and allowed deviations and the approximate of this tradeoff using similarity to expert demonstrations and deviations.

II. Background

This section presents a brief technical overview of DAgger, SafeDAgger, and different methods for approximating GPs using neural networks.

A. DAgger and SafeDAgger

The DAgger framework extends traditional supervised learning approaches by simultaneously running both an expert policy that we wish to clone and a novice policy we wish to train [14]. By aggregating new data from the expert, the underlying model and reward structure are uncovered.

Using supervised learning, we train an initial novice policy $\pi_{\text{nov},0}$ on some initial training set $\mathcal{D}_0$ generated by the expert policy π_{exp}. With this initialization, DAgger iteratively collects additional training examples from a mixture of the expert and novice policy. During a given episode, the combined expert and novice system interacts with the environment under the supervision of a decision rule. The decision rule, referred to as DR$(\cdot)$ in Algorithm 1, decides at every time-step t whether to use the action from the novice or expert to interact with the environment (Figure 2). The observations o_t received during each epoch and the expert's choice of corresponding actions make up a new dataset called $\mathcal{D}_i$. The new dataset of training examples is combined with the previous sets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$, and the novice policy is then re-trained on $\mathcal{D}$, as presented in Algorithm 1.

By allowing the novice to act, the combined system explores parts of the state space further from the nominal trajectories of the expert. In querying the expert in these parts of the state space, the novice is able to learn a more robust policy. However, allowing the novice to always act risks the possibility of encountering an unsafe state, which can be costly in real-world experiments. The VanillaDAgger algorithm and SafeDAgger balance this trade-off by their choice of decision rules.

Under VanillaDAgger (Algorithm 2), the expert's action is chosen with probability $\beta_i \in [0, 1]$, where i denotes the DAgger epoch. If $\beta_i = \lambda \beta_{i-1}$ for some $\lambda \in (0, 1)$, then the novice takes increasingly more actions each epoch. As the novice is given more training labels from previous epochs, it is allowed greater autonomy in exploring the state space. The VanillaDAgger decision-rule does not consider any similarity measure between the novice and expert actions. Hence, even if the novice suggests a highly unsafe action, VanillaDAgger allows the novice to act with probability $(1 - \beta_i)$.

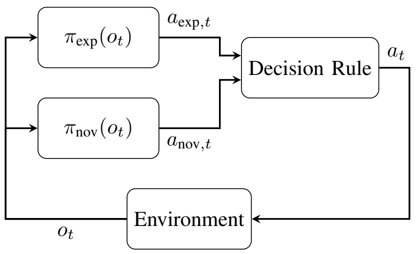


Fig. 2: Flowchart for DAgger variants, where the decision rule differs between approaches.

Algorithm 1 DAGGER

```
1: procedure DAGGER(DR(·))
2:     Initialize $\mathcal{D} \leftarrow \emptyset$
3:     Initialize $\pi_{\text{nov},i}$
4:     for epoch $i = 1 : K$
5:         Sample $T$-step trajectories using DR
6:         Get $\mathcal{D}_i = \{o_t, \pi_{\text{exp}}(o_t) \mid t \in 1 : T\}$
7:         Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
8:         Train $\pi_{\text{nov},i+1}$ on $\mathcal{D}$
```

The "optimal" decision-rule approximated by SafeDAgger, presented in Algorithm 3 and referred to as SafeDAgger*, computes the *discrepancy* between the expert and novice actions and allows the novice to act if the distance between the actions is less than some chosen threshold τ [11].[1] Though this decision rule is claimed to be optimal, we argue that it has a shortcoming.

An ideal decision rule would allow the novice to act if there is a sufficiently low probability that the system can transition to an unsafe state. If the combined system is currently near an unsafe state, the tolerable perturbation from the expert's choice of action is smaller than when the system is far from unsafe states. Hence, in practice, the

[1] To reduce the number of expert queries, SafeDAgger approximates the SafeDAgger* decision rule using a deep policy that determines whether or not the novice policy is likely to deviate from the reference policy. Unlike SafeDAgger, we are not concerned with minimizing expert queries during a given episode. Hence, we compare to the SafeDAgger* decision rule directly, as opposed to the approximation.

Algorithm 2 VANILLADAGGER Decision Rule

```
1: procedure DR($o_t, i, \beta_0, \lambda$)
2:     $a_{\text{nov},t} \leftarrow \pi_{\text{nov}}(o_t)$
3:     $a_{\text{exp},t} \leftarrow \pi_{\text{exp}}(o_t)$
4:     $\beta_i \leftarrow \lambda^i \beta_0$
5:     $z \sim \text{Uniform}(0, 1)$
6:     if $z \leq \beta_i$
7:         return $a_{\text{exp},t}$
8:     else
9:         return $a_{\text{nov},t}$
```

Algorithm 3 SAFEDAGGER* Decision Rule

```
1: procedure DR($o_t, \tau$)
2:     $a_{\text{nov},t} \leftarrow \pi_{\text{nov}}(o_t)$
3:     $a_{\text{exp},t} \leftarrow \pi_{\text{exp}}(o_t)$
4:     if $\|a_{\text{nov},t} - a_{\text{exp},t}\|^2 \leq \tau$
5:         return $a_{\text{nov},t}$
6:     else
7:         return $a_{\text{exp},t}$
```

single threshold τ employed in SafeDAgger* is either too conservative when the system is far from unsafe states or too relaxed when near them. To approximate the ideal decision rule in a model-free manner, we propose not just considering the distance between the novice's and expert's actions, but also the uncertainty in the novice policy at a given state. To estimate the uncertainty of the novice policy, we use Bayesian deep learning.

There are two works which build on the algorithm presented in this work. Kelly et al. [15] perform experiments on an autonomous vehicle and find a safe method to query humans for demonstrations, and calibrating the threshold parameters of the algorithm presented in our work. Cronrath et al. [16] propose an extension of our ideas that attempt to combine the improved safety of a Bayesian extension to DAgger with the query efficiency of SafeDAgger.

B. Bayesian Approximation Methods

Recent research has focused on approximating GPs with neural networks [17]. While GPs alone have shown great success in modeling uncertainty and approximating safety [18], traditional GP approaches are computationally expensive for high-dimensional feature spaces and large datasets [13]. Advances in deep learning have shown great success in handling these complexities. Two methods for approximating GPs with deep neural networks are ensemble methods [19] and Monte-Carlo dropout [20]. Refer to Appendix A for a summary of advantages and disadvantages of these approaches and an empirical evaluation of these methods.

In this work, we chose to use the ensemble method, which is a technique for training a collection of neural networks to execute the same task and then combining the output into a single prediction. This approach has shown to significantly improve performance in practice [21]. There is a work that employed an ensemble of neural networks to approximate GPs and demonstrated that this is a more straightforward approach to estimate predictive uncertainty [19]. Typically, neural networks predict point estimates of the output that are optimized to minimize the mean squared error on the training set. The authors claim that this approach does not capture irreducible, or *aleatoric* uncertainty, but only *epistemic* uncertainty. They propose using a *proper scoring rule*, like negative log-likelihood, as a loss function to train an ensemble in which each network predicts a mean and a variance of a Gaussian distribution over the output. They postulate that such loss functions provide a better measure of the quality of predictive uncertainty and thus reward better calibrated predictions. Network predictions are then combined as a mixture of Gaussians.

III. ENSEMBLEDAGGER

We present the EnsembleDAgger decision rule, in which the *discrepancy* between the expert's and the novice's mean action, as well as the novice's *doubt*, which is variance of the novice's action, are used to decide whether to choose the novice action. According to the EnsembleDAgger decision rule, the novice must satisfy two conditions in order to act. The first is that the discrepancy between the novice and expert's action, i.e. $\|\bar{a}_{\text{nov},t} - a_{\text{exp},t}\|^2$, must be less than some threshold τ. This is the SafeDAgger* decision rule, but will henceforth be referred to as the *discrepancy rule*. Assuming the novice policy outputs a variance on its predicted action $\sigma^2_{a_{\text{nov},t}}$, as an ensemble of neural networks would, then the second condition is that $\sigma^2_{a_{\text{nov},t}}$ is less than some threshold χ. We refer this condition as the *doubt rule*. As shown in Figure 3, in order for the novice to act according to the EnsembleDAgger decision rule, it must satisfy both the discrepncy rule and the double rule. The algorithm, described in Algorithm 4, is parameterized by the values τ and χ.

We restate the assumptions made to explain why the this decision rule is able to better guarantee the system's safety:

1) The expert prefers trajectories that avoid failure states, and rarely visits near failure states, implying that states dissimilar to those in expert trajectories (or states unfamiliar to the novice) are likely to be in closer proximity to failure states.
2) Following from (1), and by capturing epistemic uncertainty, or lack of familiarity with states in the training dataset, the novice's doubt provides a model-free proxy for proximity to failure states.
3) In order to constrain the probability of encountering a failure state, the discrepancy between the action taken and the expert's action is less than some bound.
4) The ideal bounds should be state-dependent, such that the bound is tighter in close proximity to failure states.
5) Following from (2, 4), the bound on discrepancy should decrease as the novice's doubt increases.

Also, it is assumed that the expert policy is primarily unimodal, as is commonly assumed in most imitation learning settings. Further, using a neural network based dissimilarity measure is useful for imitation learning as neural networks

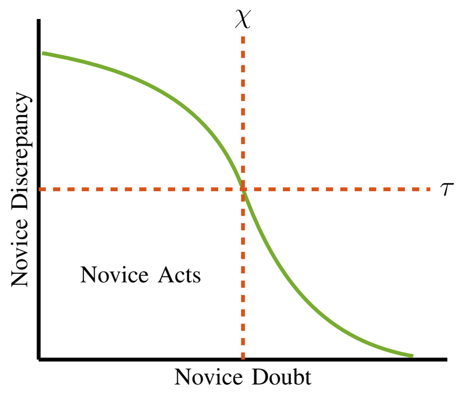


Fig. 3: The EnsembleDAgger decision rule is parametrized by doubt (χ) and discrepancy (τ) bounds, and is a low-order, model-free approximation to the 'ideal' decision rule, shown in green.

Algorithm 4 EnsembleDAgger Decision Rule

```
1: procedure DR($o_t, \tau, \chi$)
2:     $\bar{a}_{\text{nov},t}, \sigma^2_{a_{\text{nov},t}} \leftarrow \pi_{\text{nov}}(o_t)$
3:     $a_{\text{exp},t} \leftarrow \pi_{\text{exp}}(o_t)$
4:     $\hat{\tau} \leftarrow \|\bar{a}_{\text{nov},t} - a_{\text{exp},t}\|^2$
5:     $\hat{\chi} \leftarrow \sigma^2_{a_{\text{nov},t}}$
6:     if $\hat{\tau} \leq \tau$ and $\hat{\chi} \leq \chi$
7:         return $\bar{a}_{\text{nov},t}$
8:     else
9:         return $a_{\text{exp},t}$
```

scale more gracefully to high-dimensional input spaces and large datasets than most non-parametric measures.

Given that we have a measure of doubt via the variance on novice actions, we ideally would like to specify the bound on discrepancy as a monotonically increasing function of doubt. To meet this end, we have experimented with the idea of making the discrepancy bound proportional to the inverse of doubt. However, the parameters specifying an arbitrary function mapping doubt to a discrepancy bound must be considered hyperparameters to the algorithm and tuned by the practitioner. We opt for the low-order approximation to the ideal functional mapping, shown in Figure 3, because the two hyperparameters, χ and τ, are easy to interpret.

By appropriately choosing the hyper-parameters τ and χ, we satisfy the dual objectives of allowing the novice to act only if it is sufficiently confident in its action and close to the expert. As $\chi \to \infty$, the decision rule converges to that of SafeDAgger*. As $\tau \to \infty$, the decision rule ignores discrepancy, and allows the novice to act if it is confident without comparison to what the expert action is. However, since the novice is only confident in states similar to those in $\mathcal{D}$, it is likely that the novice having low doubt causes its action to also have low discrepancy, implying that the algorithm is less sensitive to an arbitrary increase in τ than to an arbitrary increase in χ. This statement is qualified in the next section by showing that using the doubt rule alone (by setting $\tau = \infty$) leads to better performance than using the discrepancy rule alone (by setting $\chi = \infty$). Though not the focus of this work, it is also worth noting that the expert only needs to be queried if the doubt rule is satisfied, thereby leading to query efficiency.

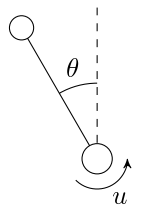


Fig. 4: The inverted pendulum environment has a state space of $[\theta, \dot{\theta}]$ and an action space of the torque u.

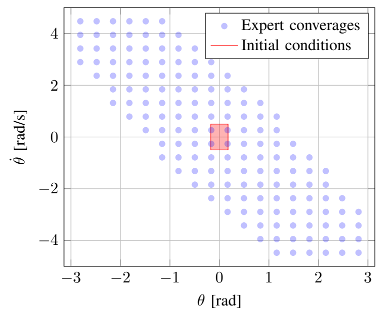


Fig. 5: This figure shows states in the expert's basin of attraction, i.e. states from which the expert converges to the origin. The figure also shows the set from which initial conditions of DAgger epochs are uniformly drawn in this experiment.

IV. Experiments

In this section, we present experimental validation for the following claims we have made:

- Using the discrepancy rule alone with fixed τ is wastefully conservative in some regions of the state space, while not conservative enough in others.
- The variance of the novice policy's output is a good measure of dissimilarity between the query state and states in the training dataset.
- Using the doubt rule alone with fixed χ trains a better performing novice policy for the same compromise to the combined (expert and novice) system's safety.
- Combining the two decision rules in the EnsembleDAgger framework improves the trained novice policy performance while making the combined system strictly safer.

To justify the first two of the above claims, we make use of a simple inverted pendulum domain. Such a simple domain is chosen because it allows us to visualize the portion of the state-space in which a given decision rule allows a given novice policy to act. We justify the latter two claims on the MuJoCo HalfCheetah OpenAI Gym environment.

A. *Inverted Pendulum domain*

Following the experimental protocol presented by Berkenkamp et al., we concretely visualize behavior by considering a deterministic but non-linear control problem of stabilizing an inverted pendulum, which has a two-dimensional state space of $[\theta, \dot{\theta}]$ and a one-dimensional action space of u, as shown in Figure 4 [22]. The control law was derived using by feedback linearization [23]. Figure 5 shows the controller's basin of attraction and highlights the states from which initial conditions are sampled uniformly during the successive epochs of DAgger. The dynamics and control law are provided in Appendix B.

The neural network model representing the novice policy is an ensemble of ten multi-layer perceptrons, each with four hidden layers of size $[64, 64, 32, 32]$ respectively. At each DAgger epoch, the ten networks are each trained for 200 training epochs with a learning rate of 10^{-3}, l_2-weight regularization of 10^{-5}, and a mini-batch size of 16. The maximum length of any trajectory is 100. No dropout or batch normalization is used. Since the data labeled by the deterministic expert is noise-free, the networks do not individually predict variance and are trained with MSE loss.

In order to compare the two decision rules, we are interested in analyzing the regions of the state space in which they allow the novice to act. We define the *permitted set* for some decision rule, given some novice and expert policies, to be the set of states in which the decision rule chooses the novice action. In Figure 6, states in the permitted set are shown as black circles. Similarly, we define the *permitted set volume* to be the fraction of states grid-sampled in $\theta \in [-\pi, \pi], \dot{\theta} \in [-5, 5]$ that are in the permitted set of a given decision rule, given some novice and expert policies. Additionally, we define the *novice basin of attraction* to be the set of states $\mathcal{X}_0$ from which, if the novice is initialized in $\mathcal{X}_0$ and allowed to act alone (without the help of the expert), the novice converges to the origin.

In order to make an apples-to-apples comparison between the two decision rules, we provide a *budget*, and analyze how the two decisions utilize this budget. The budget chosen is a fixed volume for the permitted set. At each epoch, since the novice has learned from more data, we linearly grow the permitted set volume budget. Prior to each episode, we solve for the value of χ and τ that will make the doubt and discrepancy rules respectively yield permitted sets with the desired volume. These values are found using bisection search.

The goal of this experiment is to show that, for some fixed volume permitted set, the doubt rule allocates that volume in the neighborhood of states represented in $\mathcal{D}$, justifying the claim that the novice's output variance is a good measure of dissimilarity between the query state and familiar states. Additionally, we show that the discrepancy rule haphazardly allocates volume to regions of the state space in which the novice and expert agree by chance, indicating that it is wastefully conservative in some regions of the state space, while not conservative enough in others.

In an additional experiment on this domain that can be found in the Appendix C, we compare the decision rules in a manner meaningful to a practitioner, by fixing the hyperparameters *a priori* and keeping them fixed over all epochs. For both experiments on this domain, we control the random seed specifying the initial condition for each epoch such that it varies across epoch but is the same regardless of decision rule. The trajectory followed from that initial condition will, of course, depend on the decision rule. In all experiments, as in all variants of DAgger, we initialize $\mathcal{D}$ with a zeroth epoch where only the expert is queried for the action, and the decision rule is used from the first epoch onward [14].

Figure 6 shows the evolution of the permitted set under the doubt rule and the discrepancy rule for the first three epochs of an experiment. Under the doubt rule, the permitted set is concentrated in the neighborhood of the labeled states in $\mathcal{D}$. This is because the variance of the function fitting $\mathcal{D}$ grows as we move away from labeled states, so the permitted set is constrained to be within some neighborhood of labeled states under the doubt rule. On the other hand, under the discrepancy rule, the permitted set is more haphazardly distributed over the state space with a smaller portion of the allotted volume being in the neighborhood of labeled states. We observe this because there exist arbitrary regions of the state space in which the function fitting the $\mathcal{D}$ happens to intersect the true control law purely by chance, leading to low discrepancy in these, often dangerous, regions.

We can see in Figure 6 that the trajectories resulting under the doubt rule carry the system to the edge of a familiar region of the state space, after which the expert is handed control to navigate unfamiliar regions. This behavior leads to a novice basin of attraction that is much larger than under the discrepancy rule, while no trajectories enter dangerous territory. However, under the discrepancy rule, we see that the novice is rarely allowed to carry the system away from an expert trajectory, thereby aggregating a dataset that is not much more likely to be informative than behavior cloning. This observation qualitatively suggests that the doubt rule can train a better performing novice policy for the same level of compromise to the combined (expert and novice) system's safety. This claim will be justified in the next experiment.

B. *MuJoCo HalfCheetah domain*

As stated earlier, in this experiment, we aim to demonstrate the superiority of the doubt rule over the discrepancy rule in a more complex domain with a large state and action space. Further, we show the effect of combining instances of the doubt and discrepancy rule under the EnsembleDAgger decision rule. We find that the resulting decision rule is strictly more conservative, but in some cases can lead to improvement over the component decision rules. Decision rules are compared across various settings of the hyperparameters χ and τ, which are selected *a priori* and held constant over a given experiment, unlike in the previous experiment.

The MuJoCo HalfCheetah-v1 domain (shown in Figure 7) is an OpenAI Gym environment with observations in $\mathbb{R}^{18}$ and actions in $\mathbb{R}^6$ [24]. We train an expert policy on this domain using the TRPO algorithm from the rllab codebase [25]. The goal is to learn a stable gait, with a reward for the distance from the origin reached. The purpose of this experiment is to compare the doubt rule, discrepancy rule, and a combination of the rules, in their ability to safely learn a policy that matches the expert score.

In this experiment, we use an ensemble of five neural networks, each with five hidden layers with 16-neuron widths, as the policy being trained. We use a smaller ensemble than

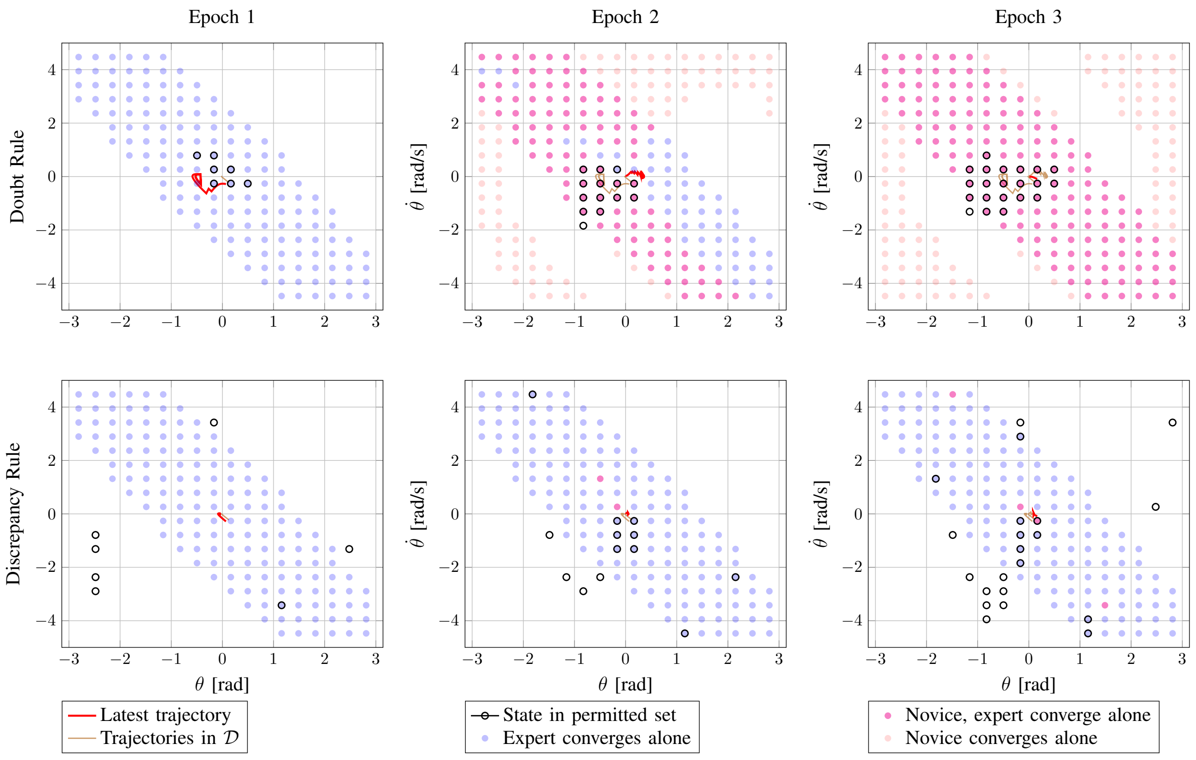


Fig. 6: Three epochs of DAgger are compared when using the doubt rule (top) and discrepancy rule (bottom). The permitted set, i.e. states at which the novice is allowed to act, denoted by black circles, of the doubt rule is concentrated in the neighborhood of states represented in $\mathcal{D}$, where as the permitted set of the discrepancy rule is distributed more haphazardly across the state space. States in which the novice alone is able to converge to the origin are indicated in pink.

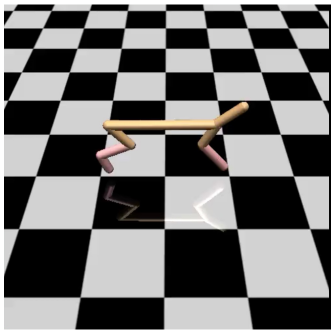


Fig. 7: The MuJoCo HalfCheetah domain.

in the last experiment to reduce simulation time. In general, the computational complexity of querying and training the novice policy is linear in the ensemble size, though larger ensembles give more accurate estimates of novice doubt.

The system is trained for seven epochs. Each epoch samples one additional trajectory of interaction with the environment, followed by re-training the novice policy on the aggregated dataset. Each trajectory is an episode with a maximum length of 100 time-steps. When training the policy on the aggregated dataset, we use a learning rate of 10^{-3}, a batch size of 32, and train for 2000 optimization epochs. Since the expert policy is stochastic, it is appropriate for the neural networks to predict the parameters of a Gaussian distribution as opposed to a point estimate of the actions. However, the networks used in this experiment simply predict point estimates since they are easier to train, and epistemic uncertainty is still captured. When training, the score over a trajectory of the lone novice and the system combined under the experiment's decision rule are queried at each epoch. Queries average the score of the policy being tested over 20 trajectories.

We define the *performance* of an instance of a given decision rule by the performance of the lone novice and the performance of the combined system, which are defined as follows. The performance of the lone novice is the average score of the novice, trained under a given decision rule instance, summed over the seven epochs of training. A better performing lone novice implies that the decision rule instance is able to quickly bring the novice to expert-level scores. Similarly, the performance of the combined system is the average score of the expert and novice, combined under a given decision rule instance, summed over the seven epochs that train the novice.

Though we have no strict notion of safety in this domain, a better performing combined system implies that trajectories perturbed under the decision rule instance are still high-scoring, are thus compromising states that one may consider to be failure states are being better avoided. We compare decision rules based on their ability to maximize the novice performance while compromising the performance of the combined system as little as possible. We sample the performance of each instance of a decision rule 100 times,

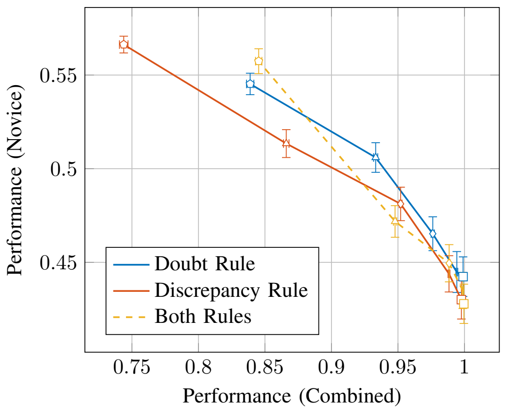


Fig. 8: Performance of various instances of the doubt rule, discrepancy rule, and combined EnsembleDAgger decision rules on the HalfCheetah domain. Consistent markers indicate the instances of the doubt and discrepancy ruled in an instance of the EnsembleDAgger decision rule.

presenting their mean and standard errors.

We test the doubt rule and the discrepancy rule using values of:

$$\begin{aligned} \chi &= [0.02, 0.05, 0.1, 0.2, 0.5] \\ \tau &= [0.2, 0.5, 1.0, 2.0, 5.0], \end{aligned}$$

and the full EnsembleDAgger decision rule at $(\tau, \chi) = [(0.2, 0.02), (0.5, 0.05), (1.0, 0.1), (2.0, 0.2), (5.0, 0.5)]$.

Though we sample (τ, χ) only on a line in the positive quadrant of $\mathbb{R}^2$, this ratio between τ and χ is chosen so the two rules are approximately equally responsible for preventing the novice action in the first training epoch.

The performance of the various decision rules for the parameters stated are shown in Figure 8 in the form of Pareto frontiers (since varying the hyperparameter trades-off between performance of the combined system and of the lone novice). We see that the doubt rule Pareto dominates the discrepancy rule, as the rule's frontier achieves better novice performance for the same compromise on the combined system's performance.

The fact that the doubt rule Pareto dominates the discrepancy rule in terms of performance is consistent with the trends observed in the inverted pendulum experiment–the doubt rule constrains the novice to act only when the state is familiar. Consequently, the perturbation from an expert action caused by choosing the novice's action is unlikely to compromise the score of the overall trajectory, though it will likely carry the system into marginally more unfamiliar territory, thereby allowing the novice to learn a more robust policy. The doubt rule, however, allows an arbitrarily large perturbation in sufficiently familiar states, and thereby can still lead to unsafe states. There exist settings of χ and τ that can make the EnsembleDAgger decision rule safe in all states, bounding the maximum perturbation from the expert action even in very familiar states. We only sample values of τ and χ along a line in $\mathbb{R}^2$, and hence do not find that points along this line show strict improvement over the independent decision rules in all cases, but see slight improvement in novice performance over the doubt rule for the case of $(\tau, \chi) = (0.5, 0.05)$. Additionally, we find the EnsembleDAgger decision to be strictly more conservative than either of the component decision rules and thus always improves the combined system performance, as expected.

V. Conclusion

In this work, we presented an extension to the DAgger algorithm that considers the safety of the novice-expert system that provides the trajectories from which the novice learns. To avoid requiring precise knowledge of safety, we assume the risk of a state to be inversely related to the size of the perturbation to an expert's action that it can accept without compromising safety. We therefore use a model-free proxy for safety by making a key assumption that the risk of a given state correlates with our familiarity with the state in the dataset $\mathcal{D}$. We expect this assumption to hold in domains where the expert is designed to maintain a margin of safety. Our algorithm replaces a weighted coin-flip that decides whether the novice acts in the VanillaDAgger decision rule. To act, the novice proposes an action that is bounded in its deviation from the expert's choice of action, as proposed by SafeDAgger* [11], but also must exhibit low variance in its choice.

In our experiments, we compared these two conditions independently, calling the first the discrepancy rule and the second the doubt rule. We found that the doubt rule effectively constrains the novice to act only in states it is familiar with, i.e. states that are within some neighborhood of states labeled in $\mathcal{D}$, while the discrepancy rule haphazardly allows the novice to act in states where there is chance agreement between their actions. Since the domain satisfies the assumptions regarding risk that are made, we find that the doubt rule is superior to the discrepancy rule in both its ability to have the novice rapidly attain expert-level control, as well as preventing the novice from carrying the combined expert-novice system into severly compromising states. Though the doubt rule alone is shown to be superior to the discrepancy rule alone, there exist hyperaparameter settings in which the conjunction of the rules is better than either individually.

Future work includes investigating methods for relaxing our risk assumptions, in particular the conflation of safety and familiarity. There exist environments with 'bottleneck' states in which the expert must frequently travel close to unsafe states to achieve its goal. Additionally, we have not provided a method for choosing the hyperparameters χ and τ, and thus intend to develop heuristic strategies that can safely discover the most suitable setting of these parameters. A recent work has already demonstrated a method for doing so on real vehicles [15].

Acknowledgments

This material is based upon work supported by SAIC Innovation Center, a subsidiary of SAIC Motors. The authors would like to acknowledge the useful feedback of Apoorva Sharma and Michael Kelly.

References

[1] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," *Available on arXiv:1606.06565*, 2016.

[2] B. Price and C. Boutilier, "Accelerating reinforcement learning through implicit imitation," *Journal of Artificial Intelligence Research*, vol. 19, pp. 569–629, 2003.

[3] S. Schaal, "Learning from demonstration," in *Advances in Neural Information Processing Systems (NIPS)*, 1997, pp. 1040–1046.

[4] J. Kober and J. Peters, "Imitation and reinforcement learning," *IEEE Robotics Automation Magazine*, vol. 17, no. 2, pp. 55–62, June 2010.

[5] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[6] H. Daumé, J. Langford, and D. Marcu, "Search-based structured prediction," *Machine Learning*, vol. 75, no. 3, pp. 297–325, 2009.

[7] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 661–668.

[8] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning (ICML)*, 2013, pp. 1–9.

[9] B. Kim and J. Pineau, "Maximum mean discrepancy imitation learning," in *Robotics: Science and Systems*, 2013.

[10] M. Laskey, S. Staszak, W. Y.-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 *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 462–469.

[11] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end autonomous driving," *Available on arXiv:1605.06450*, 2016.

[12] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with Gaussian processes," in *IEEE Conference on Decision and Control (CDC)*, 2014, pp. 1424–1431.

[13] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2005.

[14] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 627–635.

[15] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "HG-DAgger: Interactive imitation learning with human experts," *Available on arXiv:1810.02890*, 2018.

[16] C. Cronrath, E. Jorge, J. Moberg, M. Jirstrand, and B. Lennartson, "BAgger: A Bayesian algorithm for safe and query-efficient imitation learning."

[17] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as Gaussian processes," *Available on arXiv:1711.00165*, 2017.

[18] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," in *IEEE Conference on Decision and Control (CDC)*, 2016, pp. 4661–4666.

[19] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 6405–6416.

[20] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," *Available on arXiv:1506.02142*, 2015.

[21] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[22] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 908–919.

[23] H. K. Khalil, "Feedback linearization," in *Nonlinear Systems*. Upper Saddle River, N.J: Prentice Hall, 2002, ch. 13, pp. 505–508.

[24] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *Available on arXiv:1606.01540*, 2016.

[25] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning (ICML)*, 2016, pp. 1329–1338.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[27] 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 Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Appendix

A. *Bayesian Approximation Techniques*

We examine two approximations of GPs: ensemble methods and Monte-Carlo dropout.

Gal et al. propose approximating Bayesian models with neural networks trained with dropout [20]. By applying dropout at every weight layer in a network, an approximation of a Gaussian process is obtained. Given a policy trained with dropout, the network can be queried N times per input observation to obtain a distribution over actions, using randomly sampled dropout masks [20], [26].

The ensemble method is a technique for training a collection of neural networks to execute the same task and then combining the output into a single prediction. This approach has shown to significantly improve performance in practice [21]. [19] employed an ensemble of neural networks to approximate GPs and demonstrated that this is a more straightforward approach to estimate predictive uncertainty (PU). Typically, neural networks predict point estimates of the output are optimized to minimize the mean squared error on the training set. The authors claim that this does not capture *aleatoric* uncertainty, but only *epistemic* uncertainty. They propose using a *proper scoring rule*, like negative log-likelihood, as a loss function to train an ensemble in which each network predicts a mean and a variance of a Gaussian distribution over the output. They postulate that such loss functions provide a better measure of the quality of predictive uncertainty and thus reward better calibrated predictions.

1) Empirical Evaluation: To determine which approximation approach is most effective in practice, we evaluate the ability of four different methods to learn the function $f(x) = \sin(\pi x) + 0.2\sin(4\pi x)$ with only eight samples. The codebase used for the evaluation (as well as the proposed algorithms) is provided in the supplementary material.

First, we fit the function with a traditional Gaussian process to act as a baseline. We use a squared-exponential kernel with a length-scale of 10. The kernel parameters chosen are the best of nine optimizer restarts [27]. Then, we fit a ten network ensemble trained with Mean-Square-Error (MSE) loss for 300 epochs, referred to as the Vanilla Ensemble. The same ensemble is trained with Negative-Log-Likelihood (NLL) loss for 2400 epochs, so each network directly predicts uncertainty. Finally, a single network is trained using Monte-Carlo Dropout with MSE loss and a keep-probability of 75%, also trained for 2400 epochs.

All neural network models have hidden layers of size $[128, 64, 64, 64]$, respectively. ADAM with a learning rate of 10^{-3} is used to optimize the vanilla ensemble and MC-dropout, while a learning rate of 10^{-4} is used for the ensemble with predictive uncertainty. No weight regularization or batch normalization are used. We use a batch size of 4.

Each model is queried for a mean and standard deviation for its estimate of $f(x)$ for $x \in [-1.5, 1.5]$. The standard deviations of each model are scaled such that their sum matches that of the GP, for ease of visual comparison. The results of each method are shown in Figure 9.

We see that the vanilla ensemble of models and ensemble with predictive uncertainty have the most visual similarity to a GP. We also note that the vanilla ensemble achieves this performance in a small fraction of the number of epochs with which the latter two models are trained. Therefore, in our experiments, the novice neural network architecture takes the form of an ensemble of neural networks, and we train the neural network with MSE loss.

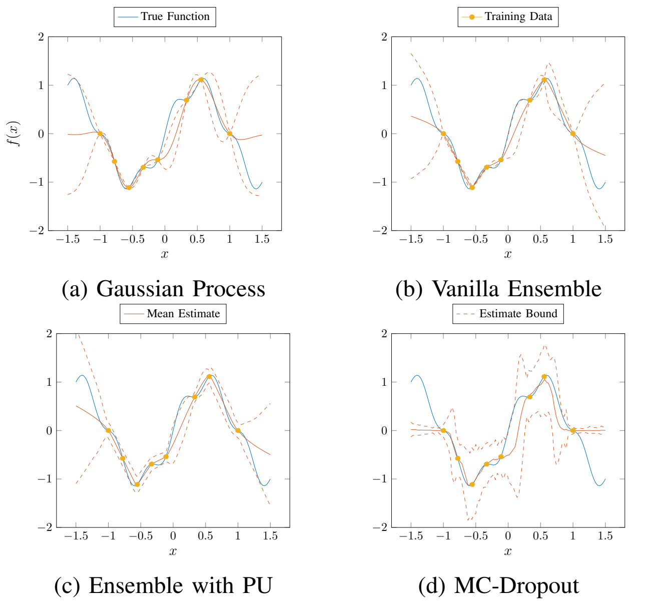


Fig. 9: Comparing Gaussian process approximation methods.

It should be noted that the implementations did not utilize adversarial training, as recommended by Lakshminarayanan et al. [19]. Additionally, the training data is noiseless, which does not highlight the potential benefit of using an ensemble with predictive uncertainty or MC-Dropout.

B. *The Inverted Pendulum domain*

The inverted pendulum has a two-dimensional state space of $[\theta, \dot{\theta}]$ and a one-dimensional action space of u, as shown in Figure 4.

The inverted pendulum is guided by the following dynamical equation:

$$\ddot{\theta} = \frac{1}{a}\sin\theta - b\dot{\theta} + cu \tag{1}$$

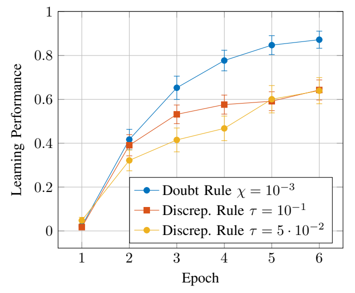
The system can be driven to the equilibrium at $[0, 0]$ by feedback linearization, using the following control law:

$$u = -\frac{a}{c}\sin\theta - \frac{1}{c}\begin{bmatrix}K_1 & K_2\end{bmatrix}\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} \tag{2}$$

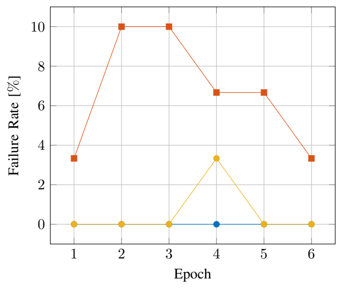
With feedback linearization, the gain vector K is computed to stabilize the linear system specified by the new dynamics:

$$\begin{bmatrix}\dot{v_1} \\ \dot{v_2}\end{bmatrix} = \begin{bmatrix}0 & 1 \\ 0 & -b\end{bmatrix}\begin{bmatrix}v_1 \\ v_2\end{bmatrix} + \begin{bmatrix}0 \\ 1\end{bmatrix} u \tag{3}$$

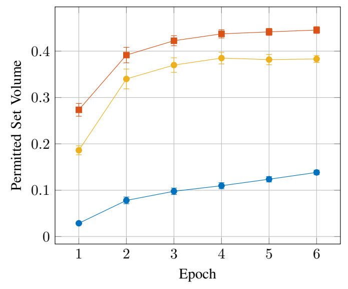
where v_1 and v_2 are the residual linear terms. The resulting controller is deterministic but sufficiently non-linear to pose an interesting learning problem. The dynamics of this environment and control law found by feedback linearization follow the example presented by Khalil et al. [23].



(a) Learning Performance



(b) Failure Rate



(c) Permitted Set Volume

Fig. 10: Comparison of the learning performance, failure rate, and permitted set volume for the doubt and discrepancy rule, where χ and τ are chosen *a priori*.

We consider the problem instance in which $a = 10$, $b = 2$, $c = 10$. Additionally, the control u is saturated to lie within $[-1, 1]$. The gains K are found to be $[0.316, 0.175]$ using a linear quadratic regulator with cost function $J = \int_0^\infty v_1^2 + v_2^2 + 10u^2$. Due to the control saturation, the controller does not converge to the desired fixed point from an arbitrary initial condition, but has the basin of attraction shown in Figure 5. Figure 5 also shows the region of the state space from which initial conditions are sampled uniformly during the successive epochs of DAgger.

C. Inverted Pendulum: Selecting Hyperparameters

In the inverted pendulum experiment discussed in the body of this paper, we solve for the hyperparameters χ and τ at every DAgger epoch such that the corresponding decision rules create a permitted set of some desired volume. Though useful for visually comparing the two decision rules, solving for the hyperparameters in this manner is not tractable in more complex problems with higher dimensional state-action spaces or with a non-deterministic expert. Hence, in our second experiment, we compare the behavior of the two decision rules in a manner more useful to a practitioner—in which the hyperparameters χ and τ are chosen *a priori*.

We introduce additional metrics of performance:

- *Learning Performance*: The fraction of states grid-sampled from $\theta \in [-\pi, \pi], \dot{\theta} \in [-5, 5]$, in both the expert and novice basin of attractions.
- *Failure Rate*: The fraction of repetitions of a given experiment in which the trajectory acquired at a given epoch, for a given decision rule and expert policy, leaves the expert's basin of attraction.

For a given choice of hyperparameters, the instances of the decision rules are compared over six epochs, and results are averaged over 30 repetitions of the experiment. During each epoch, we track the learning performance, failure rate, and permitted set volume.

Figure 10 shows the results for an instance of the doubt rule with $\chi = 10^{-3}$, the discrepancy rule with $\tau = 10^{-1}$ and the discrepancy rule with $\tau = 5 \cdot 10^{-2}$. As shown in Figure 10a, this instance of the doubt rule demonstrates superior learning performance to either instance of the discrepancy rule. In addition to demonstrating more rapid learning, this instance exhibits no failures in any of the six epochs in any of the repeated experiments, as shown in Figure 10b. Neither instance of the discrepancy rule is failure-free, and choosing the more conservative τ reduces the failure rate at the expense of learning performance.

It is also interesting to note the evolution of the permitted set volume over the six epochs for fixed hyperparameter choices. It appears that in this domain, the permitted set volumes grow monotonically, which matches expectations. However, we can see that the permitted set volumes for the discrepancy rule are many times larger than that of the doubt rule. This confirms the observations made in the previous experiment: the doubt rule is less permissive in allowing the novice to act, but is nonetheless able to generative more informative trajectories.

The results of this experiment confirm the observations made in the first inverted pendulum quantitatively. As a consequence of the discrepancy rule being too conservative in states familiar to the novice, the system is prevented from entering states that are informative to the novice's learning, and hence we see poor learning performance. Furthermore, since the discrepancy rule is not conservative enough in risky states, the discrepancy rule encounters failures more frequently than the doubt rule.