Asking for Help in Online Imitation Learning Using Support Estimation

Michael Laskey[1],Florian T. Pokorny[3], Sachin Patil[1], Ken Goldberg[2]

***Abstract*—**

I. Introduction

Consider a robot trying to learn a surgical sub task, such as cutting a circular piece of gauze. The deformable nature of the gauze makes this a complex manipulation task, where traditional control methods could fail because they assume known dynamics of the system. One way to handle this is to assume an expert is their to provide demonstrations of good behavior and have the robot to learn a controller to make sequential decisions. Imitation learning techniques have in the past achieved state of the art performance in in a variety of applications where no knowledge of a reward function or model is available [5], [2], [**?**] .

One approach to imitation was originally to train a classifier or regressor to predict an expert's behavior given training data of the encountered observations (input) and actions (output) performed by the expert. However, the agent's actions affected the next state observed, which violates the standard i.i.d assumption that is common in most statistical learning approaches. Ross et al. demonstrated how this can lead to poor performance [3]. Ross et al. then presented a state of the art algorithm known as Dagger, that at each iteration executes a trained policy then has the expert provide feedback and retrains the policy[4]. Implementations of Dagger though either don't use the expert feedback at run time or randomly does so with probability that decays exponentially per iteration. This in practice can lead the agent to deviate far from the expert policy and encounter potentially danger states [5]. We are interested in having the robot ask for help when unknown states are encountered and have the expert take over to demonstrate a proper action.

Since we are interested in the case of unknown dynamics and reward function, we only have information about which states have been visited in the past. One technique is to use a distance metric and if a new state is above a certain threshold from its nearest state then the robot should ask for help. However, this requires knowledge of what threshold to set and doesn't account for the ability of the classifier or regressor to generalize. A common belief in statistical machine learning is that a trained model will be able to generalize within the support of the distribution it is trained on. Thus, we purpose estimating the support region and asking for help when a new state leaves this region. By asking for help at each iteration, we can allow the robot to progress further along the task and potentially reduce the number of iterations needed to train a policy. Our contributions to date are framing the problem of asking for help in an online setting as estimating the support of a distribution and providing a proof for an upper bound on the expected number of times the robot's decisions differ from the expert. Initial results on a Mario AI video game averaged over 50 randomly played levels suggest a faster learning rate can be achieved over prior Dagger methods. [TODO: NEED TO GET MORE INFO FROM STEPHANE TO BE SURE ABOUT THE EMPIRICAL RESULTS].

[1]Department of Electrical Engineering and Computer Sciences; {mdlaskey, sachinpatil}@berkeley.edu

[2]Department of Industrial Engineering and Operations Research and Department of Electrical Engineering and Computer Sciences; goldberg@berkeley.edu

[1−2] University of California, Berkeley; Berkeley, CA 94720, USA

[3]Computer Vision and Active Perception Lab, Centre for Autonomous Systems, School of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm, Sweden fpokorny@kth.se

II. Preliminaries

A. Support Estimation

Support estimation can be defined as minimizing a function that contains the volume of a given probability distribution. Formally we define it as follows let $\mathbf{x}_1, ..., \mathbf{x}_l$ be i.i.d. random variables in a set $\mathcal{X}$ with distribution P. Let $\mathcal{G}$ be a class of measurable subsets of $\mathcal{X}$ and let λ be a real function defined on $\mathcal{G}$. The quantile function with respect to $(P, \lambda, \mathcal{G})$ is

$$U(\alpha) = \inf\{\lambda(G) : P(G) \geq \alpha, G \in \mathcal{G}\}\ 0 < \alpha \leq 1 \quad (1)$$

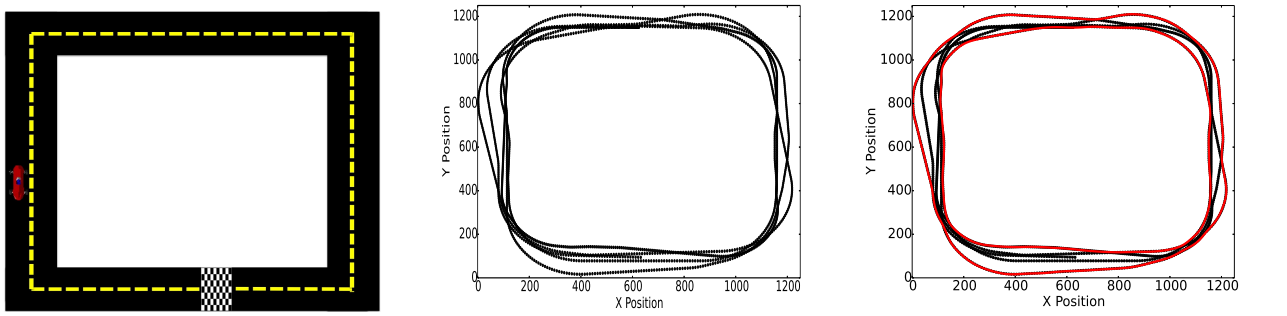
We denote by $G(\alpha)$ the $G \in \mathcal{G}$ that attains the infimum. The most common choice of λ is Lebesgue measure, in which case $G(\alpha)$ is the minimum volume $G \in \mathcal{G}$ that contains at least a fraction α of the the probability mass. Thus, $G(1)$ is the support of the density p corresponding to P, assuming it exists.

To handle high dimensional densities P, work by Scholkopf et al. looked at representing the class $\mathcal{G}$ via a kernel k as the set of half-spaces in SV feature space [6]. They then minimize a SV style regularizor which, using a kernel, controls the smoothness of the estimated function describing G. In terms of the quantile function described in Eq. 1, the approach can be thought of as employing $\lambda(G) = ||w||^2$, where $G_w = \{x : f_w(x) \geq \rho\}$. Here (w, ρ) are a weight vector and offset parameterizing a hyperplane in the feature space associated with a kernel.

Let Φ be a feature map $\mathcal{X} \to \mathcal{F}$, i.e. a map into an inner product space F such that the inner product in the image of Φ can be computed by evaluating some simple kernel

$$k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^T \Phi(\mathbf{y}) \quad (2)$$

such as the Gaussian kernel



(a) Race Car Simulator (b) Expert Demonstrations (c) Estimated Support of Expert Demonstrations

Fig. 1: A race car is driven around a track in Fig. 1(a) by an expert demonstrator. The state space of the car is represented as the x and y positions on the track or $\mathbf{s} = [x, y]$. In Fig. 1(b), the demonstrations are plotted in state space. In Fig. 1(c), the support is estimated using the decision function in 4. As illustrated the support accurately describes the boundary of the expert trajectory.

$$k(\mathbf{x}, \mathbf{y}) = e^{-||\mathbf{x}-\mathbf{y}||^2/c} \tag{3}$$

The main idea is to solve an optimization problem that separates the origin, we solve the following quadratic problem:

$$\min_{w \in F, \xi \in \mathcal{R}^l, \rho \in \mathcal{R}} \frac{1}{2}||w||^2 + \frac{1}{vl}\sum_i \xi_i - \rho$$
$$\text{s.t } (w^T \Phi(x_i)) \geq \rho - \xi_i,\ \xi_i \geq 0$$

The decision function then becomes

$$g(x) = \text{sgn}((w^T \Phi(x)) \tag{4}$$

Where $g(x) = 1$ indicates in the support region and $(g) = -1$ denotes being outside the support region. An example of this decision function can be seen in Fig. 1, where an expert drives a simulated race car around a track and the support of his demonstrations is estimated.

B. Asking for Help Using Support Estimation

Our algorithm, A.H.U.S.E. [TODO: AHUSE IS PLACEHOLDER NAME] starts our with a set of expert demonstrations denoted $\mathcal{D}$, we then train a classifier or regressor, $\hat{\pi}_0$, where $\hat{\pi} : s \rightarrow a$. Then execute the policy at each state encountered s, if $g(s) = -1$ then the expert takes over until $g(s) = 1$. The expert's actions are then added to the overall dataset and the policy is retrained. The full algorithm can be seen in Algorithm **??**.

Algorithm 1: A.H.U.S.E.

```
Initialize D ← Expert Demonstrations
Initialize π̂_1 to any policy in Π
for i=1,2,...,N do
    Execute π̂_1
    for t=1,2,...,T do
        if g(s) == 1 then
            Take Action π̂(s)_i
        else
            Take Action π*(s)
    Get dataset D_i = {(s, π*(s))} of visited states by
    π_i and actions given by expert
    Aggregate datasets: D ← D ∪ D_i
    Train π̂_{i+1} on D
    Train support estimation g(s) on D
```

III. Theoretical Analysis

We begin introducing notation relevant to our setting. We denote Π the class of policies the learner is considering the task horizon. For any policy π, we let d_π^t denote the distribution of states at time t if the learner executed policy π from the time step 1 to $t-1$. Furthermore, we denote $d_\pi = \frac{1}{T}\sum_{t=1}^{T} d_\pi^t$ the average distribution of states if we follow policy π for T steps. Given a state s, we denote $C(s, a)$ the expected immediate cost of performing action a in state s for the task we are considering and denote $C_\pi(s) = E_{a\sim\pi(s)}[C(s, a)]$ the expected immediate cost of π in s. We assume C is bound in $[0, 1]$. The total cost of executing π for T-steps (i.e. the cost-to-go) is denoted $J(\pi) = \sum_{t=1}^{T} E_{s\sim d_\pi^t}[C_\pi(s)]$.

In imitation learning, we may not necessarily know observe true costs $C(s, a)$ for the particular task. Instead we observe expert demonstrations and seek to bound $J(\pi)$ for any cost function C based on how well π mimics the expert policy's π^*. Denote l the observed surrogate loss function we minimize instead of C. For instance $l(s, \pi)$ may be the expected $0-1$ loss of π with respect to π^*. For instance $l(s, \pi)$ may be the expected $0-1$ loss of π with respect to π^* in state s, or a squared hinge loss of π with respect to π^* in s. Importantly, in many instances, C and l may be the same function-for instance, if we are interested in optimizing the learners ability to predict the actions chosen by an expert.

Our goal is to find a policy $\hat{\pi}$ which minimizes the observed surrogate loss under its induced distributions, (i.e.):

$$\hat{\pi} = \arg\min_{\pi\in\Pi} E_{s\sim d_\pi}[l(s, \pi)] \tag{5}$$

As system dynamics are assumed both unknown and complex, we cannot compute d_π and can only sample it by executing π in the system. Hence this is a non-i.i.d supervised learning problem due to the dependence of the input distribution on the policy π itself. The interaction between policy and the resulting distribution makes optimization difficult as a results in a non-convex objective even if the loss $l(s,*)$ is convex in π for all states s.

In the classification case assume we have a $l(s,\pi) = \sum_s(I(\pi(s) \neq \pi^*(s)))$, or an indicator function that is 1 if the policy is a mismatch and zero otherwise. In this case the cost function C is equivalent to matching the expert policy, thus $C(s,\pi) = l(s,\pi)$. Given a density estimation function $g(s) \rightarrow \{0,1\}$, we are interesting in bounding the cost of executing policy $\pi = g(s)\hat{\pi} + (1-g(s))\pi^*$, $J(\pi) = TE_{s\sim d_\pi, a\sim\pi^*(s)}[l(s,a,\pi)]$.

Denote the expected loss of the policy π mimicking the expert under its own distribution of states $E_{d_\pi} = E_{s\sim_{d_\pi},a\sim_{\pi^*(s)}}[l(s,a,\pi)]$.We know have the distribution the policy was trained on represented as $d_{\hat{\pi}}$ and the distribution it encounters at test time for iteration i is denoted as d_π. We now define ξ as the error in our density estimation function $g(x) \rightarrow \{0,1\}$.

The density of d_{π_i} can be defined now for some $\alpha \in [0,1]$:

$$d_{\pi_i} = (1-\alpha)d_{\hat{\pi}} + \alpha d \tag{6}$$

In this work we are also interested in interested in determining the expected loss per iteration on the policy trained $\hat{\pi}_i$ or $E_{d_{\hat{\pi}_i}} = E_{s\sim_{d_{\hat{\pi}_i}},a\sim_{\pi^*(s)}}[l(s,a,\hat{\pi}_i)]$. Let $\epsilon_N = \frac{1}{N}\sum_{n=1}^{N} E_{d_{\hat{\pi}_1}}$ or the average error encountered on the known distribution.

In order to solve this proof we must use the following lemma

Lemma 3.1: Using A.H.U.S.E., where the error in $g(x) \rightarrow \{0,1\}$ is ξ

$$E_{d_\pi} \leq E_{d_{\hat{\pi}}} + \xi \tag{7}$$

Proof:

$$\begin{aligned}
E_{d_{\pi_i}} =& \sum_x d_{\pi_i} l(s,\pi) \\
=& \sum_x ((1-\alpha)d_{\hat{\pi}} + \alpha d) l(s,\pi) \\
=& \sum_x (1-\alpha) d_{\hat{\pi}} l(s,\pi) + \sum_x \alpha d l(s,\pi) \\
=& (1-\alpha)E_{d_{\hat{\pi}}} + \alpha\xi \\
\leq& E_{d_{\hat{\pi}}} + \xi
\end{aligned}$$

■

Theorem 3.2: Using A.H.U.S.E., where the error in $g(x) \rightarrow \{0,1\}$ is ξ and the cost function $C = l$ or is upper bound by the surrogate loss function.

$$J(\pi) \leq T[\epsilon_N + \xi] \tag{8}$$

Proof:

We first demote that $J(\pi) \leq J(\bar{\pi})$, since $J(\pi) = \min_{i=1\ldots N}(T\epsilon_{\text{class}})$ and $J(\bar{\pi}) = \frac{1}{N}\sum_{i=1}^{N} J(\hat{\pi}_i)$. Since, the average is greater than or equal to the minimum.

$$\begin{aligned}
J(\bar{\pi}) =& \frac{1}{N}\sum_{i=1}^{N} J(\pi_i) \\
=& \frac{1}{N}\sum_{i=1}^{N} TE_{d_{\pi_i}} \\
\leq& \frac{1}{N}\sum_{i=1}^{N} T(E_{d_{\hat{\pi}}} + \xi) \\
\leq& T(\epsilon_N + \xi)
\end{aligned}$$

■

IV. Simulation Experiments

A. Super Mario Bros

Super Mario Bros. is a platform video game where the character, Mario, must move across each stage by avoiding being hit by enemies and falling into gaps, and before running out of time. We used the simulator from a recent Mario Bros. AI competition which can randomly stage generate stages of varying difficulty (more difficult gaps and types of enemies). Our goal is to train the computer to play the game based on current game features as input. Our expert is a near optimal A^* search algorithm that plans ahead using the game simulator. An action consists of 4 binary variables indicating which subset of buttons we should press in $\{\text{left}, \text{right}, \text{jump}, \text{speed}\}$. For both Dagger and A.H.U.S.E. we use a kernelized Support Vector Machine (SVM) as the policy representation, the kernel was chosen as the radial basis function and the bandwidth of the kernel c was set to $1/|\mathcal{D}|$, which is the default setting of sci-kit learn. We initialized both Dagger and A.H.U.S.E with two expert demonstrations for each level. For the support estimation in A.H.U.S.E. we used a radial basis function as well for the kernel with a bandwidth of $c = 0.01$.

We compare performance in terms of average distance traveled by Mario per stage before dying, running out of time or completing the stage, on randomly generated stages of difficulty 1 with a time limit of 20 s to complete the stage. The total distance of each stage is 2200. Stages of difficulty 1 are fairly easy for an average human player but contain most types of enemies and gaps, except with fewer enemies and gaps than stages of harder difficulties. We compare performance of our method and DAgger. In Fig. 3, we report the performance of distance traveled vs. the amount of time a level was tried. For A.H.U.S.E. we measure performance at how far the trained policy $\hat{\pi}$ travels, since π has an expert helping it at each iteration. The plot is averaged over 50 randomly generated levels.

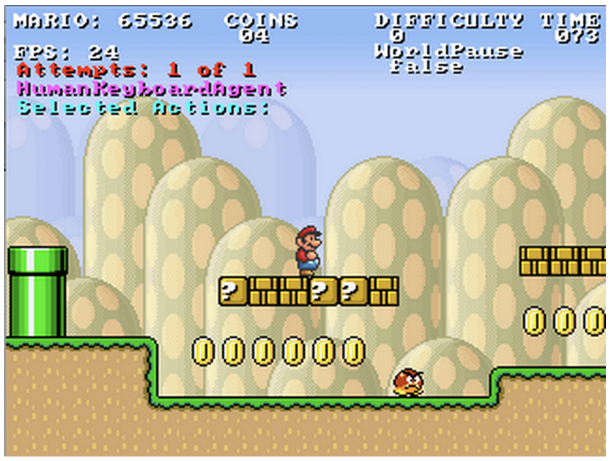


Fig. 2: An example image from the Mario AI simulator. Mario faces many challenges to reach an end of a level, such as gaps, jumps and enemies.

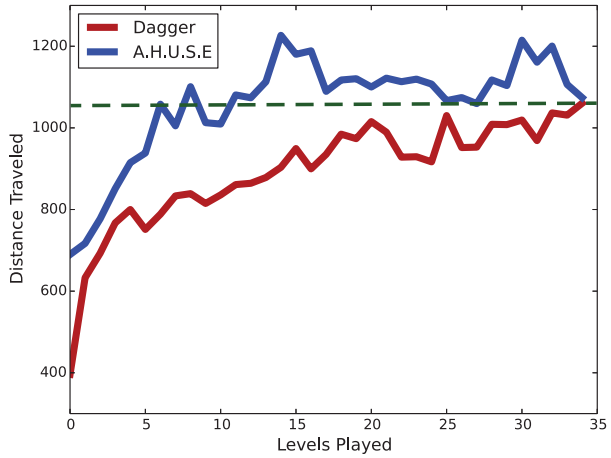


Fig. 3: The performance of distance traveled for each algorithm over trying 35 levels. For A.H.U.S.E. we measure performance at how far the trained policy $\hat{\pi}$ travels, since π always has an expert helping it. The plot is averaged over 50 randomly generated levels. [TODO: STEPHANE WAS UNCLEAR ABOUT SOME DETAILS IN THE PAPER. I WANT TO EMAIL HIM AND VERIFY EVERYTHING WITH HIM NEXT WEEK BEFORE THESE RESULTS ARE FINAL.]

REFERENCES

[1] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," DTIC Document, Tech. Rep., 1989.

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

[3] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," *arXiv preprint arXiv:1011.0686*, 2010.

[4] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1765–1772.

[5] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[6] J. Schulman, A. Gupta, S. Venkatesan, M. Tayson-Frederick, and P. Abbeel, "A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 4111–4117.