Smooth Imitation Learning

Hoang M. Le
California Institute of Technology
hmle@caltech.edu

Yisong Yue
California Institute of Technology
yyue@caltech.edu

Peter Carr
Disney Research
peter.carr@disneyresearch.com

Abstract

We study the problem of smooth imitation learning, where the goal is to train a policy that can imitate human behavior in a dynamic and continuous environment. Since such a policy will necessarily be imperfect, it should be able to smoothly recover from its mistakes. Our motivating application is training a policy to imitate an expert camera operator as she follows the action during a sport event; however our approach can be applied more generally as well. We take a learning reduction approach, where the problem of smooth imitation learning can be "reduced" to a regression problem, and the performance guarantee of the learned policy depends on the performance guarantee of the reduced regression problem (which is often much easier to analyze). Building upon previous learning reduction results, we can prove that our approach requires only a polynomial number of learning or exploration rounds before converging to a good policy. Our empirical validation confirms the efficacy and practical relevance of our approach.

1 Introduction

In many domains, one important machine learning task is to train an automated policy or controller to mimic a human expert. For example, in automated broadcasting, we require an automated camera controller that mimics the behavior of a professional camera operator in response to a dynamic environment [1]. One increasingly popular approach is to use imitation learning [3, 4], which aims to learn a policy to predict human behavior given the current state of the environment, and is essentially a "lifting" of conventional machine learning into dynamical environments.

We consider the agnostic learning setting, where any trained policy will be necessarily imperfect due to limitations in the model class and feature representation of the environment. As such, when taken to a dynamic environment, an imitation policy trained using conventional machine learning methods must account for the sequentially cascading errors caused by a drift in distribution of states at test time compared to what the model was trained on. In addition, many applications of imitation learning naturally operate in environments with very large state space and action spaces. Both these challenges can be posed as an exploration problem, i.e., how to (statistically) efficiently explore the space of possible trajectories in order to reliably train a good policy.

In this paper, we consider the problem of smooth imitation learning, which extends conventional imitation learning to the continuous regime. The goal is now to train a policy to smoothly mimic human demonstrations in a continuous and dynamic environment. This problem arises naturally in a diverse range of domains: in addition to our demonstrated application in automatic broadcasting [1], smooth imitation learning can be generally applied towards surgical robotics learning from human demonstration, learning movement of bio-prosthetic limbs using neural signals feedback, self-driving wheel-chairs, etc.

2 Problem Setup

Following the basic setup from [4], let Π denote a class of policies the learner is considering, and let T denote the time horizon of the imitation problem. Imitation learning is a sequential learning problem. In each round n, the following happens:

- Given initial state s_0 drawn from starting distribution of states, the learner executes a policy π_n, resulting a sequence of states $s_1^n, \ldots, s_T^n$.
- For each s_t^n, expert feedback $\widehat{y}_t^n$ is provided indicating what the human expert would do given s_t^n.
- The learner integrates this knowledge and proceeds to the next round $n \leftarrow n + 1$.

For any policy $\pi \in \Pi$, let d_t^π denote the distribution of states at time t if the learner executed π for the first $t-1$ time steps. Furthermore, let $d_\pi = \frac{1}{T}\sum_{t=1}^T d_t^\pi$ be the average distribution of states if we follow π for all T steps. The goal is to find a policy $\widehat{\pi} \in \Pi$ which minimizes the imitation loss under its own induced distribution of states:

$$\widehat{\pi} = \arg\min_{\pi \in \Pi} \mathbb{E}_{s \sim d_\pi} \left[\ell(s, \pi)\right], \tag{1}$$

where the (convex) loss function $\ell(s, \pi)$ captures how well π imitates expert human demonstrations under those states. One common loss is squared loss between the policy's decision and the expert demonstration: $\ell(s, \pi) = \|\widehat{y} - \pi(s)\|^2$. We assume the agnostic setting, where the minimizer of (1) does not necessarily achieve 0 loss (i.e., we cannot perfectly imitate the human expert).

For instance, in the basketball broadcasting setting, states s contain information regarding the location of the players as well as the configuration of the camera, and the policy π must decide where the point the camera next (i.e., $\pi(s)$), with the quality of that decision characterized by the loss ℓ that compares with what the human expert would do given s.

Each π_n can be thought of as an *exploration strategy* that collects labels $\widehat{y}_1^n, \ldots, \widehat{y}_T^n$ for states $s_1^n, \ldots, s_T^n$. Because of the potential branching factor of decisions, one might naively think that a very large (possibly infinite) number of learning rounds is required to fully explore the continuous state space. Previous work showed that only a polynomial number of learning rounds is required for convergence to the minimizer of (1) [3, 4], but with a dependence on the length of each round T. In the following we present an approach that removes this dependence on T for smooth imitation learning, and thus enjoys much faster convergence guarantees.

3 Our Contributions

We take a learning reduction approach, where feedback $\widehat{y}$ is integrated via standard supervised learning and can be solved by existing machine learning algorithms. A big challenge with using learning reduction is to control for the cascading errors caused by the changing dynamics of the system. As the dynamics of the system change from one policy to the next, we no longer test on the same distribution of states as during training, which violates the assumptions of supervised learning. This problem is additionally coupled with the need to explore efficiently to achieve optimal sample complexity, as we are operating in an infinite / continuous state and action space. Our reduction approach provides a solution to this coupled challenges. The key steps in the reduction are to show that:

- The empirical distribution of states that the supervised learning algorithm is trained on converges, and thus the distribution of states $d_{\widehat{\pi}}$ induced by the resulting policy (approximately) matches the distribution that $\widehat{\pi}$ was trained on.
- The learning guarantees of the supervised learning problem "lifts" to the dynamical system, and thus provides a bound on (1).

Our approach improves upon previous learning reduction approaches [3, 4] in the following ways:

- The convergence rate does not depend on the length of each round T for smooth imitation learning, and thus requires much less exploration than previous work. In addition, our approach has adaptive learning rate, thus further improves the convergence rate compared to previous approaches.
- We propose an approach to simulate a substantial amount of "virtual" expert feedback, and thus requires much fewer direct human expert demonstrations.
- Our approach is fully deterministic. Under the continuous setting, deterministic policies are strictly better than stochastic policies as (i) smoothness is critical and (ii) policy sampling requires holding more data during training, which may not be practical for infinite state and action space.

4 Approach

4.1 Learning Reduction Approach

Standard supervised regression methods assume i.i.d. training and test examples, which leads to an unsatisfactory result directly applied to sequential learning problems (e.g. left panel of figure 2). Thus, a principled reduction of smooth imitation learning to supervised regression should (approximately) preserve the i.i.d. relationship between training and test examples, and in particular the state distribution d_π should smoothly converges to a stationary distribution. With such a stable learning reduction, the performance of the regression subroutine can be used to quantify the performance of the final policy π_N (see Theorem 4.1). The key design questions are: (i) what should be the exploration policy for the next iteration? (ii) what should be the expert feedback? and (iii) how should we design a good loss function to feed into the base regression routine?

Algorithm 1 Smooth Search-Based Imitation Learning

Require: Features $\mathbf{X}$, human expert policy $\pi^\star$, base routine $Regress$, regularizer f_π
1: Initialize $\mathbf{Y}_0 = \pi^\star(\mathbf{X})$, $\mathbf{S}_0 = \phi(\mathbf{X}, \mathbf{Y}_0)$, $f_{\hat{\pi}} = \arg\min_f \|\mathbf{Y}_0 - f(\mathbf{Y}_0)\|$
2: Initial policy $\pi_0 = \hat{\pi}_0 = Regress(\mathbf{S}_0, \mathbf{Y}_0 | f_{\hat{\pi}})$.
3: **for** $n = 1, \ldots, N$ **do**
4: Roll out $\mathbf{Y}_n = \pi_{n-1}(\mathbf{S}_{n-1})$ ▷ Set exploration trajectory
5: Set exploration states $\mathbf{S}_n = \phi(\mathbf{X}, \mathbf{Y}_n)$ ▷ e.g. $s_t^n \leftarrow [x_{t:t-\tau}, y_{t-1:t-\tau}^n]$ where $y_t^n \in \mathbf{Y}_n$
6: Collect smooth expert feedback $\widehat{\mathbf{Y}}_n = \{\widehat{y}_t^n\}$ $\forall s_t^n \in \mathbf{S}_n$, ▷ Gather 1-step look-ahead feedback
7: Update regularizer $f_{\hat{\pi}}$ ▷ $f_{\hat{\pi}} = \arg\min_f \|\widehat{\mathbf{Y}}_n - f(\widehat{\mathbf{Y}}_n)\|$
8: Learn model $\hat{\pi}_n = Regress(\mathbf{S}_n, \widehat{\mathbf{Y}}_n | f_{\hat{\pi}})$
9: New policy $\pi_n = \beta\hat{\pi}_n + (1-\beta)\pi_{n-1}$ ▷ β adaptively set via loss of $\hat{\pi}_n$ vs. π_{n-1} relative to $\pi^\star$
10: **end for**
11: **return** Last policy π_N

Algorithm 1 describes our approach. The raw data $\mathcal{TS} = \{\mathbf{X}, \pi^\star\}$ consists of continuous streams of input features $\mathbf{X} = \{x_1, \ldots, x_T\}$ and a human expert demonstrator $\pi^\star$. The state space $\mathbf{S} = \phi(\mathbf{X}, \mathbf{Y})$ is defined based on both raw input $\mathbf{X}$ and an (imperfect) trajectory $\mathbf{Y}$. For example, $s_t = [x_{t:t-\tau}, y_{t-1:t-\tau}]$. The exploration trajectory $\mathbf{Y}$ of a policy π can be rolled out by sequentially applying π to $\mathbf{S}$: $y_t = \pi(s_t)$. We denote by $\widehat{\mathbf{Y}} = \{\widehat{y}_t\}$ the expert feedback actions indicating what the human expert would do given exploration states $\mathbf{S} = \{s_t\}$.

The new exploration policy for the next round (Line 4) is set as the weighted average of the previous learned regression model $\hat{\pi}_n$ and the previous exploration policy π_n (Line 9). This interpolation step plays two key roles. First, it is a form of myopic or greedy exploration that incorporates the best trained policy so far. Intuitively, rolling out π_n leads to incidental exploration on the mistakes of π_n, and so each round of exploration is focused on refining an improving policy.

Second, the interpolation in Line 9 ensures a slow drift in the distribution of states from round to round, which preserves an approximate i.i.d. property for the supervised regression subroutine and guarantees convergence of the learning reduction approach. However this model interpolation creates an inherent tension between maintaining approximate i.i.d. for valid supervised learning and more aggressive exploration (and thus faster convergence). In fact, in previous work [3], theoretical guarantees only apply for very small β ($\approx 1/T^3$). We have developed an adaptive approach for setting β that circumvents much of this tension (see Theorem 4.1), and thus guarantees a valid learning reduction while substantially reducing the rounds of exploration and learning required.

Imitation learning relies on the expert providing feedback during each roll-out of the exploration policy (Line 6). Expert feedback $\widehat{y}_t$ reflects what a human expert would have done given current (imperfect) state s_t, and the most general way to acquire $\widehat{y}_t$ is to query the expert $\pi^\star$ every time. However, in the smooth imitation regime, we can relax this requirement by allowing for a substantial amount of virtual simulation of expert feedback. The simulation can compute a smooth 1-step look-ahead correction to current state s_t based on actual, but limited human demonstrator $\pi^\star$'s response to a state. This approach can greatly reduce the need for constant human expert interactions to guide the train-

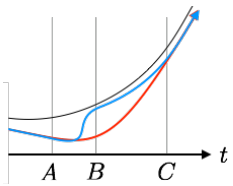

Figure 1: Red line represents a smoother simulated feedback of an imperfect trajectory compared to blue line

ing. Figure 1 depicts an example where our policy has made a mistake at Location A, and where we have a single demonstrated human trajectory from $\pi^\star$ (black line). Depending on the smoothness requirements, we can simulate virtual expert feedback as via either the blue (less smooth) or red lin (more smooth). In this work, we focus on smooth policies. See the appendix for more details.

The actual reduction is in Line 8, where a regression subroutine $Regress$ returns a newly learned policy based on current exploration states and expert feedback. In our smooth setting, the loss function of $Regress$ regularizes the smoothness of learning policy π in smooth policy class Π via an auto-regressor f_π, the parameters of which are updated at each round based on collected expert feedback. See appendix for more details.

4.2 Theoretical Guarantee

Algorithm 1 is an extension of the SEARN approach [3], and can also be interpreted as performing gradient desecnt in a smooth function space. The convergence rate of previous work [3] is tied to a small and fixed learning rate $\beta(\approx 1/T^3)$. Our approach can remove this dependency on T for policy class Π with certain self-smooth property, thus allowing for much faster convergence guarantee. We provide the following guarantee for our algorithm, the proof of which can be found in the appendix.

Theorem 4.1 (Policy Improvement)**.** *Assume the loss function ℓ is convex and Lipschitz-continuous with contant L_ℓ, policy class Π has the self-smooth property in which $\pi \in \Pi$ is Lipschitz-continuous with constant $L_\Pi < 1$, and the quality of the base regression routine is controlled by constants δ and ϵ such that $\ell(s, \widehat{\pi}(s)) \leqslant \epsilon$ and $\|\widehat{\pi}(s) - \pi(s)\| \leqslant \delta$ for all state s. We can bound the overall policy loss difference from the update rule $\pi_{new} = \beta\widehat{\pi} + (1-\beta)\pi$ in Algorithm 1 as:*

$$L(\pi_{new}) - L(\pi) \leqslant \beta\left[\epsilon - L(\pi) + (1-\beta)L_\ell L_\Pi(\delta + L_\Pi C)\right] \tag{2}$$

for $C = \max\{\delta, \frac{2\delta}{1-L_\Pi}\}$. In particular, if $\beta > 1 - \frac{L(\pi)-\epsilon}{L_\ell L_\Pi(\delta+L_\Pi C)}$, we have $L(\pi_{new}) < L(\pi)$

5 Experimental Results

We applied our method to the setting of smooth spatiotemporal prediction for realtime camera planning [1]. The motivating application is determining where a camera should look when broadcasting a sporting event. Given noisy tracking of players as raw input data, and associated camera angles from professional human operator, the learning objective is to produce a policy smoothly and accurate tracks the sporting event. Our algorithm produces policies that outperform the state-of-the-art approaches [1]. In Figure 2, we present our result (right panel), in contrast to results from supervised learning methods that ignore changing dynamics, i.e. trained with i.i.d assumption (left panel), and methods that simply apply self-smooth regularizer (smooth filter) after supervised training (middle panel). By adaptively selecting learning rate, our algorithm converges quickly to a good model after only 10 rounds of exploration.

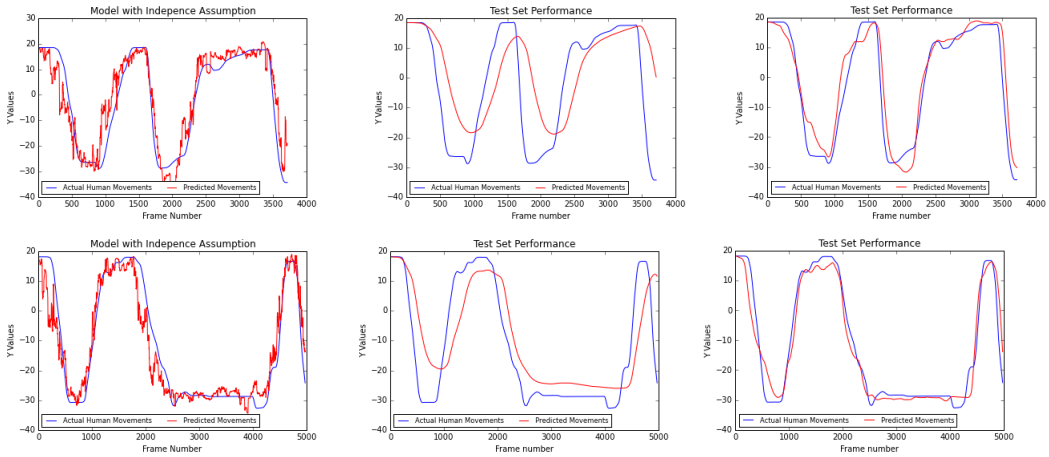

Figure 2: Left panel: Performance of standard supervised learning using independence assumption. Middle panel: Initial round, with smooth filter after learning. Right panel: Our algorithm after 10 round

References

[1] P. Carr and J. Chen. Mimicking human camera operators. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2015.

[2] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2–3):81–227, 2012.

[3] H. Daumé III, J. Langford, and D. Marcu. Search-based structured prediction. *Machine Learning*, 75(3):297–325, 2009.

[4] S. Ross, G. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

A Virtual Feedback, Loss Function and Smooth Decision Tree Regression

A.1 Virtual Simulation of Expert Feedback

In the absence of human expert during training, we can simulate the interactive expert feedback based on existing (but limited) human demonstration trajectory $\mathbf{Y}^* = \pi^\star(\mathbf{X})$. During training, we roll out an imperfect exploration policy π to obtain the exploration trajectory $\mathbf{Y}$ and associated exploration states $\mathbf{S} = \phi(\mathbf{X}, \mathbf{Y})$. Since policy is imperfect, exploration action y_t is potentially far off from real human action y_t^* given raw input x_t. We can simulate an interactive feedback $\widehat{y}_t$ that is a smooth recovery from current state s_t such that $\widehat{y}_t$ is a smooth transition from previous time steps along exploration trajectory $y_{t-1:t-\tau}$, while moving closer to the correct human actions $y_t^*, y_{t+1}^*, \ldots$. There are multiple ways to computationally simulate the expert feedback to satisfy this objective. In our implementation, we chose the expert correction to be

$$\widehat{y}_t = y_t^* + e^{-\lambda}(y_t - y_t^*)$$

where y_t^* is the human response to input x_t, y_t is the current exploration action, and the parameter $\lambda > 0$ dictates how aggressively the simulated expert is recovering the demonstrated human trajectory. (i.e. larger λ implies less smooth recovery). Intuitively, if at time t we allow the simulated expert takes over the exploration trajectory, this formulation will let the simulated expert to converge to real human trajectory exponentially fast, albeit at different degrees of smoothness depending on parameter λ. Note that various other computational regimes will work to simulate the expert. The only strict requirement is that the simulated feedback $\widehat{y}_t$ should inch closer to ground truth y_t^* compared to imperfect exploration action y_t. However, smooth correction of expert is important for the stability and smoothness of the new learning model.

A.2 Loss Function Design

For an exploration policy π with corresponding rolled-out trajectory $\mathbf{Y} = \{y_t\}_{t=1}^T$, we form exploration states $\mathbf{S} = \phi(\mathbf{X}, \mathbf{Y})$ and collect expert feedback $\widehat{\mathbf{Y}}$. A base regression routine is then called to learn a new model (line 8 of algorithm 1). The loss function used for this regression should satisfy the dual goal of smoothness and accuracy. We approximate the smoothness of the curve by a smooth auto-regressor f_π that satisfies $y_t \approx f_\pi\left(y_{t-1:t-\tau}\right)$. In the loss function f_π acts as a smooth regularizer, the parameters of which can be updated at each around based on expert feedback $\widehat{\mathbf{Y}}$ (line 7 of algorithm 1) according to $f_{\widehat{\pi}} = \arg\min_f(\|\widehat{\mathbf{Y}} - f(\widehat{\mathbf{Y}})\|)$. The regression routine should optimize the trade-off between $y_t \approx \widehat{y}_t$ (expert label) versus smoothness as dictated by regularizer f_π. With regularization parameters defined for $f_{\widehat{\pi}}$, the base regression routine will train a new policy π using the joint loss function $\mathcal{L}(\pi) = L_D(\pi) + wL_S(\pi) = \|\mathbf{Y} - \widehat{\mathbf{Y}}\|^2 + w\|\mathbf{Y} - f_{\widehat{\pi}}(\mathbf{Y})\|^2$. Here w is a hyper-parameter that controls how much we care about smoothness versus absolute accuracy relative to expert trajectory $\widehat{\mathbf{Y}}$.

The nature of f_π varies depending on application domains. In our broadcasting example, our smoothness regularizer f_π is a linear auto-regressor based on previous τ frames, where τ is the number of time steps with which an imperfect trajectory can be smoothly recovered by an expert's correction. In this setting, f_π is parameterized by a set of smoothness coefficient $c_\pi = [c_1, \ldots, c_\tau]$ such that given a trajectory $\mathbf{Y}$, c_π is the minimizer of smoothness loss $L_S(\mathbf{Y}) = \sum_{t=1}^T(y_t - \sum_{i=1}^\tau(c_i y_{t-i}))^2$ (via least squares fit), and $f_\pi = f_\pi\left(y_{t-1:t-\tau}|c_\pi\right) = \sum_{i=1}^\tau c_i y_{t-i}$. With the smooth coefficients c_π determined, the base regression routine will train a new policy π using the joint loss function $\mathcal{L}(\pi) = L_D(\pi) + wL_S(\pi) = \sum_{t=1}^T(y_t - \widehat{y}_t)^2 + w\sum_{t=1}^T(y_t - \sum_{i=1}^\tau c_i y_{t-1})^2$, where $\widehat{y}_t$ is the expert feedback at time t, y_t is the trajectory the regression routine needs to learn, and w is a pre-set parameter that trade-off smoothness versus absolute accuracy. In the next section, we develop an extension of traditional decision tree-based ensembles to accommodate this joint loss function.

A.3 Smooth Regression Tree

Empirically, decision tree-based ensembles are among the best performing supervised machine learning method [2]. Due to the piece-wise constant nature of decision tree-based prediction, the results are inevitably non-smooth. We provide an extension to classical decision tree-based regression, where prediction at leaf node is not necessarily a constant, but is a function of both static leaf

node prediction and input features. Let the generic input and output space be $\mathcal{I}$ and $\mathcal{O}$ respectively. We denote and decouple the input features to the predictor by $\phi = \langle u, v \rangle \in \mathcal{I}$ where u and v are both (multi-dimensional) vectors, but v is the vector of *dependent* input features that influences the prediction. Let the generic output value be y. For the ease of presentation, we view $y \in \mathbb{R}^1$. However, the framework can easily be generalized into multi-dimensional output space. We have a training data set $\mathcal{TS} = \{(\langle u_t, v_t \rangle, y_t)_{t=1}^N\}$. In the traditional decision tree setting, the algorithm learns a function $T : \mathcal{I} \mapsto \mathcal{O}$ such that at test time, given a new input to the predictor $\phi_{test} = \langle u_{test}, v_{test} \rangle$, T would take $\langle u_{test}, v_{test} \rangle$ to navigate to a terminal leaf node that contains a subset of training data $\mathcal{P} \subset \mathcal{TS}$ and outputs a constant (average) prediction $y_{predict} = \frac{1}{|\mathcal{P}|} \sum_{(\phi_t, y_t) \in \mathcal{P}} y_t$. We extend this framework such that the prediction made at terminal leaf node is a function of both the static prediction and dependent input features v_{new}. This framework is appealing as the base regression routine in our algorithms should handle a loss function that depends on both output values and certain input features.

Recall that in traditional regression tree, the training goal is to predict $\hat{y}_t = \arg\min_y \mathcal{L}_{\mathcal{TS}}(y)$ from input ϕ_t such that to minimize the global loss $\mathcal{L}_{\mathcal{TS}}(y) = \sum_t (y - y_t)^2 = \sum_t D(y, y_t)$ where D is the usual squares distance loss. In our new setting, we want to additionally approximate $\hat{y}_t \approx f(v_t)$ on top of the usual objective $\hat{y}_t \approx y_t$. We modify the loss function to incorporate both objectives and seek to minimize instead $\mathcal{L}_{\mathcal{TS}}(y) = \sum_t D(y, y_t) + S(y_t, z_t)$ where we have $f(v_t) = \arg\min_y S(y, v_t)$. A natural choice is to simply set D and S to be the squared loss function to yield closed-form solutions. In our broadcasting application, we choose $D(y, y_t) = (y - y_t)^2$ and smoothness loss $S(y_t, z_t) = w(y_t - f(z_t))^2$, yielding $\mathcal{L}_{\mathcal{TS}}(y) = \sum_t (y - y_t)^2 + w(y_t - f(z_t))^2$, where w is a hyper-parameter that controls how much we care about the loss given by S relative to D.

Setting terminal node value. Given a terminal leaf node with training data $\mathcal{P} \subset \mathcal{TS}$, we want to set a node value $\bar{y}_{node}$ such that

$$\begin{aligned} \bar{y}_{node} &= \arg\min_y \mathcal{L}_{\mathcal{P}}(y) = \arg\min_y \sum_{(\phi_t, y_t) \in \mathcal{P}} D(y, y_t) + S(y_t, v_t) \\ &= \arg\min_y \sum_{(\phi_t, y_t) \in \mathcal{P}} (y - y_t)^2 + w(y_t - f(v_t))^2 = \frac{\sum_{(\phi_t, y_t) \in \mathcal{P}} y_t}{|\mathcal{P}|} \end{aligned} \tag{3}$$

which is the simple average of output values within subset $\mathcal{P}$. Note that this is not necessarily the same as terminal node values in traditional decision trees due to the presence of $S(y_t, v_t)$ during splitting.

Making Prediction. Let input to the predictor at test time be $\phi_{test} = \langle u_{test}, v_{test} \rangle$. We use input features u_{test} and v_{test} to navigate to a terminal node, representing by subset $\mathcal{P}$ of the training data. The prediction $y_{predict}$ is made such that

$$\begin{aligned} y_{predict} &= \arg\min_y D(y, \bar{y}_{node}) + S(y, v_{test}) = \arg\min_y (y - \bar{y}_{node})^2 + w(y - f(v_{test}))^2 \\ &= \frac{\bar{y}_{node} + w f(v_{test})}{1 + w} \end{aligned}$$

where $\bar{y}_{node}$ is set according to equation (3).

Training and Node Splitting Mechanism: For a node representing a subset $\mathcal{P}$ of the training data, the node impurity is defined as:

$$I_{node} = \mathcal{L}_{\mathcal{P}}(\bar{y}_{node}) = \sum_{(\phi_t, y_t) \in \mathcal{P}} D(\bar{y}_{node}, y_t) + S(y_t, v_t) = \sum_{(\phi_t, y_t) \in \mathcal{P}} \left[(\bar{y}_{node} - y_t)^2 + w(y_t - f(v_t))^2 \right]$$

where $\bar{y}_{node}$ is set according to equation (3) over (ϕ_t, y_t)'s in $\mathcal{P}$. At each possible splitting point where $\mathcal{P}$ is partitioned into $\mathcal{P}_{left}$ and $\mathcal{P}_{right}$, the impurity of the left and right child of the node is defined similarly. As with normal decision tree, the best splitting point is chosen as one that maximizes the impurity reduction: $I_{node} - \frac{|\mathcal{P}_{left}|}{|\mathcal{P}|} I_{left} - \frac{|\mathcal{P}_{right}|}{|\mathcal{P}|} I_{right}$

B Theoretical Analysis

In this section, we provide the proof to theorem 4.1. Let T be the trajectory horizon. For a policy π in the deterministic policy class Π, given a starting state s_0, we roll out the full trajectory $s_0 \xrightarrow{\pi} s_1 \xrightarrow{\pi} \dots \xrightarrow{\pi} s_T$. Let $\ell(s, a)$ be the loss of taking action a at state s, we can define the trajectory loss of policy π from starting state s_0 as

$$L(\pi|s_0) = \frac{1}{T}\sum_{t=1}^{T} \ell(s_t, \pi(s_t))$$

For a starting state distribution $\mathcal{D}$, we define policy loss of π as $L(\pi) = \mathbb{E}_{s_0\sim\mathcal{D}}[L(\pi|s_0)]$. To simplify notations, we define $s_t = [x_t, \pi(s_{t-1})]$ where x_t encodes the featurized input at current time step, and $\pi(s_{t-1})$ encodes the dependency on previous predictions. Our results easily extend to the case where s_t depends on previous τ predictions. We skip the subscript to consider general policy update rule within each iteration

$$\pi' = \pi_{new} = \beta\widehat{\pi} + (1-\beta)\pi \tag{4}$$

where π is the current policy (combined up until the previous iteration), $\widehat{\pi}$ is the learned model from calling the base regression routine $Regress(\mathbf{S}, \widehat{\mathbf{Y}}|f_{\widehat{\pi}})$. Interpolation parameter β is adaptively chosen in each iteration. We are interested in quantifying the policy improvement when updating π to π'. Specifically, we want to bound

$$\Delta = L(\pi') - L(\pi)$$

Note that Searn [3] and Dagger [4] provide bounds for Δ that are generally positive, meaning the policy does not degrade too much after each update. In our analysis, we point out choices of β where learning policies can strictly improve.

Based on update rule (4), consider rolling out π' and π from the same starting state s_0 to obtain two separate trajectories $\pi' \longmapsto [s_0 \to s'_1 \dots \to s'_T]$ and $\pi \longmapsto [s_0 \to s_1 \dots \to s_T]$. We will bound the loss difference of old and new policies starting from same state s_0

$$\Delta(s_0) = \frac{1}{T}\sum_{t=1}^{T} \ell(s'_t, \pi'(s'_t)) - \ell(s_t, \pi(s_t))$$

Assume convexity of ℓ (e.g. sum of square losses):

$$\begin{aligned}\ell(s'_t, \pi'(s'_t)) &= \ell(s'_t, \beta\widehat{\pi}(s'_t) + (1-\beta)\pi(s'_t)) \\ &\leqslant \beta\ell(s'_t, \widehat{\pi}(s'_t)) + (1-\beta)\ell(s'_t, \pi(s'_t))\end{aligned}$$

Thus we can begin to bound individual components of $\Delta(s_0)$ as

$$\ell(s'_t, \pi'(s'_t)) - \ell(s_t, \pi(s_t)) \leqslant \beta\ell(s'_t, \widehat{\pi}(s'_t)) + (1-\beta)\ell(s'_t, \pi(s'_t)) - \ell(s_t, \pi(s_t))$$

Let the upperbound on loss of learned model $\widehat{\pi}$ be ϵ such that $\forall t$: $\ell(s'_t, \widehat{\pi}(s'_t)) \leqslant \epsilon$, we then have:

$$\Delta(s_0) \leqslant \beta\epsilon - \beta L(\pi|s_0) + (1-\beta)\frac{1}{T}\sum_{t=1}^{T}\left[\ell(s'_t, \pi(s'_t)) - \ell(s_t, \pi(s_t))\right] \tag{5}$$

$$\leqslant \beta\epsilon - \beta L(\pi|s_0) + (1-\beta)\frac{1}{T}\sum_{t=1}^{T} L_\ell\|\pi(s'_t) - \pi(s_t)\| \tag{6}$$

for L_ℓ-Lipschitz loss function ℓ.

Recall that $s'_t = [x_t, \pi'(s'_{t-1})]$ and $s_t = [x_t, \pi(s_{t-1})]$, assume the self-smooth Lipschitz property of policy class Π with Lipschitz constant $L_\Pi < 1$, we have:

$$\begin{aligned}\|\pi(s'_t) - \pi(s_t)\| &= \|\pi([x_t, \pi'(s'_{t-1})]) - \pi([x_t, \pi(s_{t-1})]\| \\ &\leqslant L_\Pi\|\pi'(s'_{t-1}) - \pi(s_{t-1})\|\end{aligned}$$

Combine this with inequality (6), we have a bound for Δ at s_0 as:

$$\Delta(s_0) \leqslant \beta\epsilon - \beta L(\pi|s_0) + (1-\beta)L_\ell L_\Pi\frac{1}{T}\sum_{t=0}^{T-1}\|\pi'(s'_t) - \pi(s_t)\| \tag{7}$$

For any state s, let the upperbound on the quality of the regression routine $Regress$ controlled by δ, i.e. $\forall s$, $\|\widehat{\pi}(s) - \pi(s)\| \leqslant \delta$. Using triangle inequality, we obtain:

$$\|\pi'(s'_t) - \pi(s_t)\| \leqslant \|\pi'(s'_t) - \pi(s'_t)\| + \|\pi(s'_t) - \pi(s_t)\| \tag{8}$$

$$= \beta\|\widehat{\pi}(s'_t) - \pi(s'_t)\| + \|\pi(s'_t) - \pi(s_t)\| \tag{9}$$

$$\leqslant \beta\delta + L_\Pi\|s'_t - s_t\| \tag{10}$$

Given a policy class Π with $L_\Pi < 1$, the following claim can be proved by induction:
Claim: For $C = \max\{\delta, \frac{2\delta}{1-L_\Pi}\}$, we have $\|s'_t - s_t\| \leqslant \beta C$

Proof. Induction on t □

Combine the above claim with inequalities (10) and (7), we have

$$\|\pi'(s'_t) - \pi(s_t)\| \leqslant \beta\delta + L_\Pi\beta C \quad \text{and} \tag{11}$$

$$\Delta(s_0) \leqslant \beta\epsilon - \beta L(\pi|s_0) + (1-\beta)L_\ell L_\Pi(\beta\delta + L_\Pi\beta C) \tag{12}$$

Integrating (12) over starting state s_0 and rearrange, we arrive at the following policy improvement bound:

$$L(\pi_{new}) - L(\pi) = L(\pi') - L(\pi) \leqslant \beta\left[\delta - L(\pi) + (1-\beta)L_\ell L_\Pi(\epsilon + L_\Pi C)\right] \tag{13}$$

This means in the worst case, as we choose $\beta \to 0$, we have $L(\pi') - L(\pi) \to 0$, meaning the new policy does not degrade much, and if we choose $\beta > 1 - \frac{L(\pi)-\delta}{L_\ell L_\Pi(\epsilon + L_\Pi C)}$, we obtain a strictly better policy as $L(\pi') < L(\pi)$.