Smooth Imitation Learning for Online Sequence Prediction

Hoang M. Le HMLE@CALTECH.EDU
Andrew Kang AKANG@CALTECH.EDU
Yisong Yue YYUE@CALTECH.EDU
California Institute of Technology, Pasadena, CA, USA

Peter Carr PETER.CARR@DISNEYRESEARCH.COM
Disney Research, Pittsburgh, PA, USA

Abstract

We study the problem of smooth imitation learning for online sequence prediction, where the goal is to train a policy that can smoothly imitate demonstrated behavior in a dynamic and continuous environment in response to online, sequential context input. Since the mapping from context to behavior is often complex, we take a learning reduction approach to reduce smooth imitation learning to a regression problem using complex function classes that are regularized to ensure smoothness. We present a learning meta-algorithm that achieves fast and stable convergence to a good policy. Our approach enjoys several attractive properties, including being fully deterministic, employing an adaptive learning rate that can provably yield larger policy improvements compared to previous approaches, and the ability to ensure stable convergence. Our empirical results demonstrate significant performance gains over previous approaches.

1. Introduction

In many complex planning and control tasks, it can be very challenging to explicitly specify a good policy. For such tasks, the use of machine learning to automatically learn a good policy from observed expert behavior, also known as imitation learning or learning from demonstrations, has proven tremendously useful (Abbeel & Ng, 2004; Ratliff et al., 2009; Argall et al., 2009; Ross & Bagnell, 2010; Ross et al., 2011; Jain et al., 2013).

In this paper, we study the problem of imitation learning for smooth online sequence prediction in a continuous regime. Online sequence prediction is the problem of making online decisions in response to exogenous input from the environment, and is a special case of reinforcement learning (see Section 2). We are further interested in policies that make smooth predictions in a continuous action space.

Our motivating example is the problem of learning smooth policies for automated camera planning (Chen et al., 2016): determining where a camera should look given environment information (e.g., noisy person detections) and corresponding demonstrations from a human expert.[1] It is widely accepted that a smoothly moving camera is essential for generating aesthetic video (Gaddam et al., 2015). From a problem formulation standpoint, one key difference between smooth imitation learning and conventional imitation learning is the use of a "smooth" policy class (which we formalize in Section 2), and the goal now is to mimic expert demonstrations by choosing the best smooth policy.

The conventional supervised learning approach to imitation learning is to train a classifier or regressor to predict the expert's behavior given training data comprising input/output pairs of contexts and actions taken by the expert. However, the learned policy's prediction affects (the distribution of) future states during the policy's actual execution, and so violates the crucial i.i.d. assumption made by most statistical learning approaches. To address this issue, numerous learning reduction approaches have been proposed (Daumé III et al., 2009; Ross & Bagnell, 2010; Ross et al., 2011), which iteratively modify the training distribution in various ways such that any supervised learning guarantees provably lift to the sequential imitation setting (potentially at the cost of statistical or computational efficiency).

We present a learning reduction approach to smooth imitation learning for online sequence prediction, which we call SIMILE (**S**mooth **IMI**tation **LE**arning). Building

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

[1] Access data at http://www.disneyresearch.com/publication/smooth-imitation-learning/ and code at http://github.com/hoangminhle/SIMILE.

upon learning reductions that employ policy aggregation (Daumé III et al., 2009), we provably lift supervised learning guarantees to the smooth imitation setting and show much faster convergence behavior compared to previous work. Our contributions can be summarized as:

- We formalize the problem of smooth imitation learning for online sequence prediction, and introduce a family of smooth policy classes that is amenable to supervised learning reductions.
- We present a principled learning reduction approach, which we call SIMILE. Our approach enjoys several attractive practical properties, including learning a fully deterministic stationary policy (as opposed to SEARN (Daumé III et al., 2009)), and not requiring data aggregation (as opposed to DAgger (Ross et al., 2011)) which can lead to super-linear training time.
- We provide performance guarantees that lift the the underlying supervised learning guarantees to the smooth imitation setting. Our guarantees hold in the agnostic setting, i.e., when the supervised learner might not achieve perfect prediction.
- We show how to exploit a stability property of our smooth policy class to enable adaptive learning rates that yield provably much faster convergence compared to SEARN (Daumé III et al., 2009).
- We empirically evaluate using the setting of smooth camera planning (Chen et al., 2016), and demonstrate the performance gains of our approach.

2. Problem Formulation

Let $\mathbf{X} = \{x_1, \ldots, x_T\} \subset \mathcal{X}^T$ denote a context sequence from the environment $\mathcal{X}$, and $\mathbf{A} = \{a_1, \ldots, a_T\} \subset \mathcal{A}^T$ denote an action sequence from some action space $\mathcal{A}$. Context sequence is exogenous, meaning a_t does not influence future context x_{t+k} for $k \geqslant 1$. Let Π denote a policy class, where each $\pi \in \Pi$ generates an action sequence $\mathbf{A}$ in response to a context sequence $\mathbf{X}$. Assume $\mathcal{X} \subset \mathbb{R}^m, \mathcal{A} \subset \mathbb{R}^k$ are continuous and infinite, with $\mathcal{A}$ non-negative and bounded such that $\vec{\mathbf{0}} \preceq a \preceq R\vec{\mathbf{1}} \;\; \forall a \in \mathcal{A}$.

Predicting actions a_t may depend on recent contexts $x_t, \ldots, x_{t-p}$ and actions $a_{t-1}, \ldots, a_{t-q}$. Without loss of generality, we define a state space $\mathcal{S}$ as $\{s_t = [x_t, a_{t-1}]\}$.[2] Policies π can thus be viewed as mapping states $\mathcal{S} = \mathcal{X} \times \mathcal{A}$ to actions $\mathcal{A}$. A roll-out of π given context sequence $\mathbf{X} = \{x_1, \ldots, x_T\}$ is the action sequence $\mathbf{A} = \{a_1, \ldots, a_T\}$:

$$\begin{aligned} a_t &= \pi(s_t) = \pi([x_t, a_{t-1}]), \\ s_{t+1} &= [x_{t+1}, a_t] \quad \forall t \in [1, \ldots, T]. \end{aligned}$$

Note that unlike the general reinforcement learning problem, we consider the setting where the state space splits into external and internal components (by definition, a_t influences subsequent states s_{t+k}, but not x_{t+k}). The use of exogenous contexts $\{x_t\}$ models settings where a policy needs to take online, sequential actions based on external environmental inputs, e.g. smooth self-driving vehicles for obstacle avoidance, helicopter aerobatics in the presence of turbulence, or smart grid management for external energy demand. The technical motivation of this dichotomy is that we will enforce smoothness only on the internal state.

Consider the example of autonomous camera planning for broadcasting a sport event (Chen et al., 2016). $\mathcal{X}$ can correspond to game information such as the locations of the players, the ball, etc., and $\mathcal{A}$ can correspond to the pan-tilt-zoom configuration of the broadcast camera. Manually specifying a good camera policy can be very challenging due to sheer complexity involved with mapping $\mathcal{X}$ to $\mathcal{A}$. It is much more natural to train $\pi \in \Pi$ to mimic observed expert demonstrations. For instance, Π can be the space of neural networks or tree-based ensembles (or both).

Following the basic setup from (Ross et al., 2011), for any policy $\pi \in \Pi$, let d_t^π denote the distribution of states at time t if π is executed 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 of imitation learning is to find a policy $\hat{\pi} \in \Pi$ which minimizes the imitation loss under its own induced distribution of states:

$$\hat{\pi} = \operatorname*{argmin}_{\pi \in \Pi} \ell_\pi(\pi) = \operatorname*{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim d_\pi}\left[\ell(\pi(s))\right], \tag{1}$$

where the (convex) imitation loss $\ell(\pi(s))$ captures how well π imitates expert demonstrations for state s. One common ℓ is squared loss between the policy's decision and the expert demonstration: $\ell(\pi(s)) = \|\pi(s) - \pi^*(s)\|^2$ for some norm $\|.\|$. Note that computing ℓ typically requires having access to a training set of expert demonstrations π^* on some set of context sequences. We also assume an agnostic setting, where the minimizer of (1) does not necessarily achieve 0 loss (i.e. it cannot perfectly imitate the expert).

2.1. Smooth Imitation Learning & Smooth Policy Class

In addition to accuracy, a key requirement of many continuous control and planning problems is smoothness (e.g., smooth camera trajectories). Generally, "smoothness" may reflect domain knowledge about stability properties or approximate equilibria of a dynamical system. We thus formalize the problem of *smooth imitation learning* as minimizing (1) over a smooth policy class Π.

Most previous work on learning smooth policies focused on simple policy classes such as linear models (Abbeel & Ng, 2004), which can be overly restrictive. We instead define a much more general smooth policy class Π as a regularized space of complex models.

Definition 2.1 (Smooth policy class Π). *Given a complex model class $\mathcal{F}$ and a class of smooth regularizers $\mathcal{H}$, we define smooth policy class $\Pi \subset \mathcal{F} \times \mathcal{H}$ as satisfying:*

[2] We can always concatenate consecutive contexts and actions.

$$\Pi \triangleq \{\pi = (f,h), f \in \mathcal{F}, h \in \mathcal{H} \mid \pi(s) \textit{ is close to both } f(x,a) \textit{ and } h(a) \ \forall \textit{ induced state } s = [x,a] \in \mathcal{S}\}$$

where closeness is controlled by regularization.

For instance, $\mathcal{F}$ can be the space of neural networks or decision trees and $\mathcal{H}$ be the space of smooth analytic functions. Π can thus be viewed as policies that predict close to some $f \in \mathcal{F}$ but are regularized to be close to some $h \in \mathcal{H}$. For sufficiently expressive $\mathcal{F}$, we often have that $\Pi \subset \mathcal{F}$. Thus optimizing over Π can be viewed as constrained optimization over $\mathcal{F}$ (by $\mathcal{H}$), which can be challenging. Our SIMILE approach integrates alternating optimization (between $\mathcal{F}$ and $\mathcal{H}$) into the learning reduction. We provide two concrete examples of Π below.

Example 2.1 (Π_λ)**.** Let $\mathcal{F}$ be any complex supervised model class, and define the simplest possible $\mathcal{H} \triangleq \{h(a) = a\}$. Given $f \in \mathcal{F}$, the prediction of a policy π can be viewed as regularized optimization over the action space to ensure closeness of π to both f and h:

$$\pi(x,a) = \operatorname*{argmin}_{a' \in \mathcal{A}} \left\|f(x,a) - a'\right\|^2 + \lambda \left\|h(a) - a'\right\|^2 = \frac{f(x,a) + \lambda h(a)}{1+\lambda} = \frac{f(x,a) + \lambda a}{1+\lambda}, \tag{2}$$

where regularization parameter λ trades-off closeness to f and to previous action. For large λ, $\pi(x,a)$ is encouraged make predictions that stays close to previous action a.

Example 2.2 (Linear auto-regressor smooth regularizers)**.** Let $\mathcal{F}$ be any complex supervised model class, and define $\mathcal{H}$ using linear auto-regressors, $\mathcal{H} \triangleq \{h(a) = \theta^\top a\}$, which model actions as a linear dynamical system (Wold, 1939). We can define π analogously to (2).

In general, SIMILE requires that Π satisfies a smooth property stated below. This property, which is exploited in our theoretical analysis (see Section 5), is motivated by the observation that given a (near) constant stream of context sequence, a stable behavior policy should exhibit a corresponding action sequence with low curvature. The two examples above satisfy this property for sufficiently large λ.

Definition 2.2 (H-state-smooth imitation policy)**.** For small constant $0 < H \ll 1$, a policy $\pi([x,a])$ is H-state-smooth if it is H-smooth w.r.t. a, i.e. for fixed $x \in \mathcal{X}$, $\forall a, a' \in \mathcal{A}$, $\forall i$: $\left\|\nabla\pi^i([x,a]) - \nabla\pi^i([x,a'])\right\|_* \leqslant H\left\|a - a'\right\|$ where π^i indicates the i^{th} component of vector-valued function[3] $\pi(s) = \left[\pi^1(s), \ldots, \pi^k(s)\right] \in \mathbb{R}^k$, and $\|.\|$ and $\|.\|_*$ are some norm and dual norm respectively. For twice differentiable policy π, this is equivalent to having the bound on the Hessian $\nabla^2\pi^i([x,a]) \preceq H\mathbb{I}_k \ \ \forall i$.

3. Related Work

The most popular traditional approaches for learning from expert demonstration focused on using approximate policy iteration techniques in the MDP setting (Kakade & Langford, 2002; Bagnell et al., 2003). Most prior approaches operate in discrete and finite action space (He et al., 2012; Ratliff et al., 2009; Abbeel & Ng, 2004; Argall et al., 2009). Some focus on continuous state space (Abbeel & Ng, 2005), but requires a linear model for the system dynamics. In contrast, we focus on learning complex smooth functions within continuous action and state spaces.

One natural approach to tackle the more general setting is to reduce imitation learning to a standard supervised learning problem (Syed & Schapire, 2010; Langford & Zadrozny, 2005; Lagoudakis & Parr, 2003). However, standard supervised methods assume i.i.d. training and test examples, thus ignoring the distribution mismatch between training and rolled-out trajectories directly applied to sequential learning problems (Kakade & Langford, 2002). Thus a naive supervised learning approach normally leads to unsatisfactory results (Ross & Bagnell, 2010).

Iterative Learning Reductions. State-of-the-art learning reductions for imitation learning typically take an iterative approach, where each training round uses standard supervised learning to learn a policy (Daumé III et al., 2009; Ross et al., 2011). In each round n, the following happens:

- Given initial state s_0 drawn from the starting distribution of states, the learner executes current policy π_n, resulting in a sequence of states $s_1^n, \ldots, s_T^n$.
- For each s_t^n, a label $\widehat{a}_t^n$ (e.g., expert feedback) is collected indicating what the expert would do given s_t^n, resulting in a new dataset $\mathcal{D}_n = \{(s_t, \widehat{a}_t^n)\}$.
- The learner integrates $\mathcal{D}_n$ to learn a policy $\hat{\pi}_n$. The learner updates the current policy to π_{n+1} based on $\hat{\pi}_n$ and π_n.

The main challenge is controlling for the cascading errors caused by the changing dynamics of the system, i.e., the distribution of states in each $\mathcal{D}_n \sim d_{\pi_n}$. A policy trained using d_{π_n} induces a different distribution of states than d_{π_n}, and so is no longer being evaluated on the same distribution as during training. A principled reduction should (approximately) preserve the i.i.d. relationship between training and test examples. Furthermore the state distribution d_π should converge to a stationary distribution.

The arguably most notable learning reduction approaches for imitation learning are SEARN (Daumé III et al., 2009) and DAgger (Ross et al., 2011). At each round, SEARN learns a new policy $\hat{\pi}_n$ and returns a distribution (or mixture) over previously learned policies: $\pi_{n+1} = \beta\hat{\pi}_n + (1-\beta)\pi_n$ for $\beta \in (0,1)$. For appropriately small choices of β, this stochastic mixing limits the "distribution drift" between π_n and π_{n+1} and can provably guarantee that the

[3] This emphasizes the possibility that π is a vector-valued function of a. The gradient and Hessian are viewed as arrays of k gradient vectors and Hessian matrices of 1-d case, since we simply treat action in $\mathbb{R}^k$ as an array of k standard functions.

performance of π_{n+1} does not degrage significantly relative to the expert demonstrations.[4]

DAgger, on the other hand, achieves stability by aggregating a new dataset at each round to learn a new policy from the combined data set $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_n$. This aggregation, however, significantly increases the computational complexity and thus is not practical for large problems that require many iterations of learning (since the training time grows super-linearly w.r.t. the number of iterations).

Both SEARN and DAgger showed that only a polynomial number of training rounds is required for convergence to a good policy, but with a dependence on the length of horizon T. In particular, to non-trivially bound the total variation distance $\|d_{\pi_{new}} - d_{\pi_{old}}\|_1$ of the state distributions between old and new policies, a learning rate $\beta < \frac{1}{T}$ is required to hold (Lemma 1 of Daumé III, Langford, and Marcu (2009) and Theorem 4.1 of Ross, Gordon, and Bagnell (2011)). As such, systems with very large time horizons might suffer from very slow convergence.

Our Contributions. Within the context of previous work, our SIMILE approach can be viewed as extending SEARN to smooth policy classes with the following improvements:

- We provide a policy improvement bound that does not depend on the time horizon T, and can thus converge much faster. In addition, SIMILE has adaptive learning rate, which can further improve convergence.
- For the smooth policy class described in Section 2, we show how to generate simulated or "virtual" expert feedback in order to guarantee stable learning. This alleviates the need to have continuous access to a dynamic oracle / expert that shows the learner what to do when it is off-track. In this regard, the way SIMILE integrates expert feedback subsumes the set-up from SEARN and DAgger.
- Unlike SEARN, SIMILE returns fully deterministic policies. 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 spaces.
- Our theoretical analysis reveals a new sequential prediction setting that yields provably fast convergence, in particular for smooth policy classes on finite-horizon problems. Existing settings that enjoy such results are limited to Markovian dynamics with discounted future rewards or linear model classes.

4. Smooth Imitation Learning Algorithm

Our learning algorithm, called SIMILE (**S**mooth **IMI**tation **LE**arning), is described in Algorithm 1. At a high level, the process can be described as:

Algorithm 1 SIMILE (**S**mooth **IMI**tation **LE**arning)

```
Input: features $\mathbf{X} = \{x_t\}$, human trajectory $\mathbf{A}^* = \{a_t^*\}$,
    base routine Train, smooth regularizers $h \in \mathcal{H}$
1:  Initialize $\mathbf{A}_0 \leftarrow \mathbf{A}^*, \mathbf{S}_0 \leftarrow \{[x_t, a_{t-1}^*]\}$,
        $h_0 = \operatorname*{argmin}_{h\in\mathcal{H}} \sum_{t=1}^{T} \|a_t^* - h(a_{t-1}^*)\|$
2:  Initial policy $\pi_0 = \hat{\pi}_0 \leftarrow \texttt{Train}(\mathbf{S}_0, \mathbf{A}_0 |\ h_0)$
3:  for $n = 1, \ldots, N$ do
4:      $\mathbf{A}_n = \{a_t^n\} \leftarrow \pi_{n-1}(\mathbf{S}_{n-1})$    //sequential roll-out
5:      $\mathbf{S}_n \leftarrow \{s_t^n = [x_t, a_{t-1}^n]\}$  //$s_t^n = [x_{t:t-p}, a_{t-1:t-q}]$
6:      $\widehat{\mathbf{A}}_n = \{\widehat{a}_t^n\}\ \forall s_t^n \in \mathbf{S}_n$    // collect smooth feedback
7:      $h_n = \operatorname*{argmin}_{h\in\mathcal{H}} \sum_{t=1}^{T} \|\widehat{a}_t^n - h(\widehat{a}_{t-1}^n)\|$ //new regularizer
8:      $\hat{\pi}_n \leftarrow \texttt{Train}(\mathbf{S}_n, \widehat{\mathbf{A}}_n |\ h_n)$    // train policy
9:      $\beta \leftarrow \beta(\ell(\hat{\pi}_n), \ell(\pi_{n-1}))$    //adaptively set $\beta$
10:     $\pi_n = \beta\hat{\pi}_n + (1-\beta)\pi_{n-1}$    // update policy
11: end for
output Last policy $\pi_N$
```

1. Start with some initial policy $\hat{\pi}_0$ (Line 2).
2. At iteration n, use π_{n-1} to build a new state distribution $\mathbf{S}_n$ and dataset $\mathcal{D}_n = \{(s_t^n, \widehat{a}_t^n)\}$ (Lines 4-6).
3. Train $\hat{\pi}_n = \operatorname{argmin}_{\pi\in\Pi} \mathbb{E}_{s\sim\mathbf{S}_n}[\ell_n(\pi(s))]$, where ℓ_n is the imitation loss (Lines 7-8). Note that ℓ_n needs not be the original ℓ, but simply needs to converge to it.
4. Interpolate $\hat{\pi}_n$ and π_{n-1} to generate a new deterministic policy π_n (Lines 9-10). Repeat from Step 2 with $n \leftarrow n+1$ until some termination condition is met.

Supervised Learning Reduction. The actual reduction is in Lines 7-8, where we follow a two-step procedure of first updating the smooth regularize h_n, and then training $\hat{\pi}_n$ via supervised learning. In other words, `Train` finds the best $f \in \mathcal{F}$ possible for a fixed h_n. We discuss how to set the training targets $\widehat{a}_t^n$ below.

Policy Update. The new policy π_n is a deterministic interpolation between the previous π_{n-1} and the newly learned $\hat{\pi}_n$ (Line 10). In contrast, for SEARN, π_n is a stochastic interploation (Daumé III et al., 2009). Lemma 5.2 and Corollary 5.3 show that deterministic interpolation converges at least as fast as stochastic for smooth policy classes.

This interpolation step plays two key roles. First, it is a form of myopic or greedy online learning. Intuitively, rolling out π_n leads to incidental exploration on the mistakes of π_n, and so each round of training is focused on refining π_n. Second, the interpolation in Line 10 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.

However this model interpolation creates an inherent tension between maintaining approximate i.i.d. for valid su-

[4] A similar approach was adopted in Conservative Policy Iteration for the MDP setting (Kakade & Langford, 2002).

pervised learning and more aggressive exploration (and thus faster convergence). For example, SEARN's guarantees only apply for small $\beta < 1/T$. SIMILE circumvents much of this tension via a policy improvement bound that allows β to adaptively increase depending on the quality of $\hat{\pi}_n$ (see Theorem 5.6), which thus guarantees a valid learning reduction while substantially speeding up convergence.

Feedback Generation. We can generate training targets $\widehat{a}_t^n$ using "virtual" feedback from simulating expert demonstrations, which has two benefits. First, we need not query the expert π^* at every iteration (as done in DAgger (Ross et al., 2011)). Continuously acquiring expert demonstrations at every round can be seen as a special case and a more expensive strategy. Second, virtual feedback ensures stable learning, i.e., every $\hat{\pi}_n$ is a feasible smooth policy.

Consider Figure 1, where our policy π_n (blue/red) made a mistake at location A, and where we have only a single expert demonstration from π^* (black). Depending on the smoothness requirements of the policy class, we can simulate virtual expert feedback as via either the red line (more smooth) or blue (less smooth) as a tradeoff between squared imitation loss and smoothness.

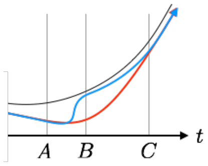


Figure 1.

When the roll-out of π_{n-1} (i.e. $\mathbf{A}_n$) differs substantially from $\mathbf{A}^*$, especially during early iterations, using smoother feedback (red instead of blue) can result in more stable learning. We formalize this notion for Π_λ in Proposition 5.8. Intuitively, whenever π_{n-1} makes a mistake, resulting in a "bad" state s_t^n, the feedback should recommend a smooth correction $\widehat{a}_t^n$ w.r.t. $\mathbf{A}_n$ to make training "easier" for the learner.[5] The virtual feedback $\widehat{a}_t^n$ should converge to the expert's action a_t^*. In practice, we use $\widehat{a}_t^n = \sigma a_t^n + (1-\sigma)a_t^*$ with $\sigma \to 0$ as n increases (which satisfies Proposition 5.8).

5. Theoretical Results

All proofs are deferred to the supplementary material.

5.1. Stability Conditions

One natural smoothness condition is that $\pi([x,a])$ should be stable w.r.t. a if x is fixed. Consider the camera planning setting: the expert policy π^* should have very small curvature, since constant inputs should correspond to constant actions. This motivates Definition 2.2, which requires that Π has low curvature given fixed context. We also show that smooth policies per Definition 2.2 lead to stable actions, in the sense that "nearby" states are mapped to "nearby" actions. The following helper lemma is useful:

Lemma 5.1. *For a fixed x, define $\pi([x,a]) \triangleq \varphi(a)$. If φ is non-negative and H-smooth w.r.t. a., then:*

$$\forall a, a' : \left(\varphi(a) - \varphi(a')\right)^2 \leqslant 6H\left(\varphi(a) + \varphi(a')\right)\left\|a - a'\right\|^2.$$

Writing π as $\pi([x,a]) \triangleq [\pi^1([x,a]), \ldots, \pi^k([x,a])]$ with each $\pi^i([x,a])$ H-smooth, Lemma 5.1 implies $\|(\pi([x,a]) - \pi([x,a']))\| \leqslant \sqrt{12HR}\,\|a-a'\|$ for R upper bounding $\mathcal{A}$. Bounded action space means that a sufficiently small H leads to the following stability conditions:

Condition 1 (Stability Condition 1)**.** Π *satisfies the Stability Condition 1 if for a fixed input feature x, the actions of π in states $s = [x,a]$ and $s' = [x,a']$ satisfy $\|\pi(s) - \pi(s')\| \leqslant \|a - a'\|$ for all $a, a' \in \mathcal{A}$.*

Condition 2 (Stability Condition 2)**.** Π *satisfies Stability Condition 2 if each π is γ-Lipschitz continuous in the action component a with $\gamma < 1$. That is, for a fixed x the actions of π in states $s = [x,a]$ and $s' = [x,a']$ satisfy $\|\pi(s) - \pi(s')\| \leqslant \gamma\,\|a - a'\|$ for all $a, a' \in \mathcal{A}$.*

These two conditions directly follow from Lemma 5.1 and assuming sufficiently small H. Condition 2 is mildly stronger than Condition 1, and enables proving much stronger policy improvement compared to previous work.

5.2. Deterministic versus Stochastic

Given two policies π and $\hat{\pi}$, and interpolation parameter $\beta \in (0,1)$, consider two ways to combine policies:

1. **stochastic**: $\pi_{sto}(s) = \hat{\pi}(s)$ with probability β, and $\pi_{sto}(s) = \pi(s)$ with probability $1-\beta$
2. **deterministic**: $\pi_{det}(s) = \beta\hat{\pi}(s) + (1-\beta)\pi(s)$

Previous learning reduction approaches only use stochastic interpolation (Daumé III et al., 2009; Ross et al., 2011), whereas SIMILE uses deterministic. The following result shows that deterministic and stochastic interpolation yield the same expected behavior for smooth policy classes.

Lemma 5.2. *Given any starting state s_0, sequentially execute π_{det} and π_{sto} to obtain two separate trajectories $\boldsymbol{A} = \{a_t\}_{t=1}^T$ and $\tilde{\boldsymbol{A}} = \{\tilde{a}_t\}_{t=1}^T$ such that $a_t = \pi_{det}(s_t)$ and $\tilde{a}_t = \pi_{sto}(\tilde{s}_t)$, where $s_t = [x_t, a_{t-1}]$ and $\tilde{s}_t = [x_t, \tilde{a}_{t-1}]$. Assuming the policies are stable as per Condition 1, we have $\mathbb{E}_{\tilde{\boldsymbol{A}}}[\tilde{a}_t] = a_t \ \ \forall t = 1, \ldots, T$, where the expectation is taken over all random roll-outs of π_{sto}.*

Lemma 5.2 shows that deterministic policy combination (SIMILE) yields unbiased trajectory roll-outs of stochastic policy combination (as done in SEARN & CPI). This represents a major advantage of SIMILE, since the number of stochastic roll-outs of π_{sto} to average to the deterministic trajectory of π_{det} is polynomial in the time horizon T, leading to much higher computational complexity. Furthermore, for convex imitation loss $\ell_\pi(\pi)$, Lemma 5.2 and Jensen's inequality yield the following corollary, which states that under convex loss, deterministic policy performs at least no worse than stochastic policy in expectation:

[5] A similar idea was proposed (He et al., 2012) for DAgger-type algorithm, albeit only for linear model classes.

Corollary 5.3 (Deterministic Policies Perform Better)**.** *For deterministic π_{det} and stochastic π_{sto} interpolations of two policies π and $\hat{\pi}$, and convex loss ℓ, we have:*

$$\begin{aligned}\ell_{\pi_{det}}(\pi_{det}) &= \ell_{\pi_{sto}}(\mathbb{E}[\pi_{sto}]) \\ &\leqslant \mathbb{E}\left[\ell_{\pi_{sto}}(\pi_{sto})\right]\end{aligned}$$

where the expectation is over all roll-outs of π_{sto}.

Remark. We construct a simple example to show that Condition 1 may be necessary for iterative learning reductions to converge. Consider the case where contexts $\mathbf{X} \subset \mathbb{R}$ are either constant or vary neglibly. Expert demonstrations should be constant $\pi^*([x_n, a^*]) = a^*$ for all n. Consider an unstable policy π such that $\pi(s) = \pi([x,a]) = ka$ for fixed $k > 1$. The rolled-out trajectory of π *diverges* π^* at an exponential rate. Assume optimistically that $\hat{\pi}$ learns the correct expert behavior, which is simply $\hat{\pi}(s) = \hat{\pi}([x,a]) = a$. For any $\beta \in (0,1)$, the updated policy $\pi' = \beta\hat{\pi} + (1-\beta)\pi$ becomes $\pi'([x,a]) = \beta a + (1-\beta)ka$. Thus the sequential roll-out of the new policy π' will also yield an exponential gap from the correct policy. By induction, the same will be true in all future iterations.

5.3. Policy Improvement

Our policy improvement guarantee builds upon the analysis from SEARN (Daumé III et al., 2009), which we extend to using adaptive learning rates β. We first restate the main policy improvement result from Daumé III et al. (2009).

Lemma 5.4 (SEARN's policy nondegradation - Lemma 1 from Daumé III et al. (2009))**.** *Let $\ell_{max} = \sup_{\pi,s} \ell(\pi(s))$, π' is defined as π_{sto} in lemma 5.2. Then for $\beta \in (0, 1/T)$:*

$$\ell_{\pi'}(\pi') - \ell_\pi(\pi) \leqslant \beta T \mathbb{E}_{s\sim d_\pi}\left[\ell(\hat{\pi}(s))\right] + \frac{1}{2}\beta^2 T^2 \ell_{max}.$$

SEARN guarantees that the new policy π' does not degrade from the expert π^* by much only if $\beta < 1/T$. Analyses of SEARN and other previous iterative reduction methods (Ross et al., 2011; Kakade & Langford, 2002; Bagnell et al., 2003; Syed & Schapire, 2010) rely on bounding the variation distance between d_π and $d_{\pi'}$. Three drawbacks of this approach are: (i) non-trivial variation distance bound typically requires β to be inversely proportional to time horizon T, causing slow convergence; (ii) not easily applicable to the continuous regime; and (iii) except under MDP framework with discounted infinite horizon, previous variation distance bounds do not guarantee monotonic policy improvements (i.e. $\ell_{\pi'}(\pi') < \ell_\pi(\pi)$).

We provide two levels of guarantees taking advantage of Stability Conditions 1 and 2 to circumvent these drawbacks. Assuming the Condition 1 and convexity of ℓ, our first result yields a guarantee comparable with SEARN.

Theorem 5.5 (T-dependent Improvement)**.** *Assume ℓ is convex and L-Lipschitz, and Condition 1 holds. Let $\epsilon = \max_{s\sim d_\pi} \|\hat{\pi}(s) - \pi(s)\|$. Then:*

$$\ell_{\pi'}(\pi') - \ell_\pi(\pi) \leqslant \beta\epsilon L T + \beta\left(\ell_\pi(\hat{\pi}) - \ell_\pi(\pi)\right). \quad (3)$$

In particular, choosing $\beta \in (0, 1/T)$ yields:

$$\ell_{\pi'}(\pi') - \ell_\pi(\pi) \leqslant \epsilon L + \beta\left(\ell_\pi(\hat{\pi}) - \ell_\pi(\pi)\right). \quad (4)$$

Similar to SEARN, Theorem 5.5 also requires $\beta \in (0, 1/T)$ to ensure the RHS of (4) stays small. However, note that the reduction term $\beta(\ell_\pi(\hat{\pi}) - \ell_\pi(\pi))$ allows the bound to be strictly negative if the policy $\hat{\pi}$ trained on d_π significantly improves on $\ell_\pi(\pi)$ (i.e., guaranteed policy improvement). We observe empirically that this often happens, especially in early iterations of training.

Under the mildly stronger Condition 2, we remove the dependency on the time horizon T, which represents a much stronger guarantee compared to previous work.

Theorem 5.6 (Policy Improvement)**.** *Assume ℓ is convex and L-Lipschitz-continuous, and Condition 2 holds. Let $\epsilon = \max_{s\sim d_\pi} \|\hat{\pi}(s) - \pi(s)\|$. Then for $\beta \in (0,1)$:*

$$\ell_{\pi'}(\pi') - \ell_\pi(\pi) \leqslant \frac{\beta\gamma\epsilon L}{(1-\beta)(1-\gamma)} + \beta(\ell_\pi(\hat{\pi}) - \ell_\pi(\pi)). \quad (5)$$

Corollary 5.7 (Monotonic Improvement)**.** *Following the notation from Theorem 5.6, let $\Delta = \ell_\pi(\pi) - \ell_\pi(\hat{\pi})$ and $\delta = \frac{\gamma\epsilon L}{1-\gamma}$. Then choosing step size $\beta = \frac{\Delta-\delta}{2\Delta}$, we have:*

$$\ell_{\pi'}(\pi') - \ell_\pi(\pi) \leqslant -\frac{(\Delta-\delta)^2}{2(\Delta+\delta)}. \quad (6)$$

The terms ϵ and $\ell_\pi(\hat{\pi}) - \ell_\pi(\pi)$ on the RHS of (4) and (5) come from the learning reduction, as they measure the "distance" between $\hat{\pi}$ and π on the state distribution induced by π (which forms the dataset to train $\hat{\pi}$). In practice, both terms can be empirically estimated from the training round, thus allowing an estimate of β to minimize the bound.

Theorem 5.6 justifies using an adaptive and more aggressive interpolation parameter β to update policies. In the worst case, setting β close to 0 will ensure the bound from (5) to be close to 0, which is consistent with SEARN's result. More generally, monotonic policy improvement can be guaranteed for appropriate choice of β, as seen from Corollary 5.7. This strict policy improvement was not possible under previous iterative learning reduction approaches such as SEARN and DAgger, and is enabled in our setting due to exploiting the smoothness conditions.

5.4. Smooth Feedback Analysis

Smooth Feedback Does Not Hurt: Recall from Section 4 that one way to simulate "virtual" feedback for training a new $\hat{\pi}$ is to set the target $\hat{a}_t = \sigma a_t + (1-\sigma)a_t^*$ for $\sigma \in (0,1)$, where smooth feedback corresponds to $\sigma \to 1$. To see that simulating smooth "virtual" feedback target does not hurt the training progress, we alternatively view SIMILE as performing gradient descent in a smooth function space (Mason et al., 1999). Define the cost functional $C : \Pi \to \mathbb{R}$ over policy space to be the average imitation loss over $\mathcal{S}$ as $C(\pi) = \int_{\mathcal{S}} \|\pi(s) - \pi^*(s)\|^2 \, dP(s)$. The gradient (Gâteaux derivative) of $C(\pi)$ w.r.t. π is:

$$\nabla C(\pi)(s) = \frac{\partial C(\pi + \alpha\delta_s)}{\partial \alpha}\Big|_{\alpha=0} = 2(\pi(s) - \pi^*(s)),$$

where δ_s is Dirac delta function centered at s. By first order approximation $C(\pi') = C(\beta\hat{\pi} + (1-\beta)\pi) = C(\pi + \beta(\hat{\pi} - \pi)) \approx C(\pi) + \beta\langle\nabla C(\pi), \hat{\pi} - \pi\rangle$. Like traditional gradient descent, we want to choose $\hat{\pi}$ such that the update moves the functional along the direction of negative gradient. In other words, we want to learn $\hat{\pi} \in \Pi$ such that $\langle\nabla C(\pi), \hat{\pi} - \pi\rangle \ll 0$. We can evaluate this inner product along the states induced by π. We thus have the estimate:

$$\langle\nabla C(\pi), \hat{\pi} - \pi\rangle \approx \frac{2}{T}\sum_{t=1}^{T}(\pi(s_t) - \pi^*(s_t))(\hat{\pi}(s_t) - \pi(s_t))$$

$$= \frac{2}{T}\sum_{t=1}^{T}(a_t - a_t^*)(\hat{\pi}([x_t, a_{t-1}]) - a_t).$$

Since we want $\langle\nabla C(\pi), \hat{\pi} - \pi\rangle < 0$, this motivates the construction of new data set $\mathcal{D}$ with states $\{[x_t, a_{t-1}]\}_{t=1}^T$ and labels $\{\widehat{a}_t\}_{t=1}^T$ to train a new policy $\hat{\pi}$, where we want $(a_t - a_t^*)(\widehat{a}_t - a_t) < 0$. A sufficient solution is to set target $\widehat{a}_t = \sigma a_t + (1-\sigma)a_t^*$ (Section 4), as this will point the gradient in negative direction, allowing the learner to make progress.

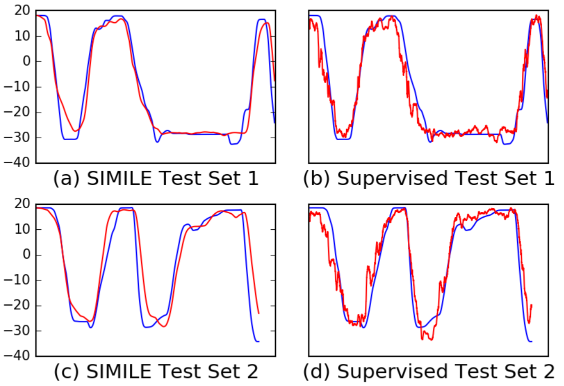
Smooth Feedback is Sometimes Necessary: When the current policy performs poorly, smooth virtual feedback may be required to ensure stable learning, i.e. producing a feasible smooth policy at each training round. We formalize this notion of feasibility by considering the smooth policy class Π_λ in Example 2.1. Recall that smooth regularization of Π_λ via $\mathcal{H}$ encourages the next action to be close to the previous action. Thus a natural way to measure smoothness of $\pi \in \Pi_\lambda$ is via the average first order difference of consecutive actions $\frac{1}{T}\sum_{t=1}^T \|a_t - a_{t-1}\|$. In particular, we want to explicitly constrain this difference relative to the expert trajectory $\frac{1}{T}\sum_{t=1}^T \|a_t - a_{t-1}\| \leqslant \eta$ at each iteration, where $\eta \propto \frac{1}{T}\sum_{t=1}^T \|a_t^* - a_{t-1}^*\|$.

When π performs poorly, i.e. the "average gap" between current trajectory $\{a_t\}$ and $\{a_t^*\}$ is large, the training target for $\hat{\pi}$ should be lowered to ensure learning a smooth policy is feasible, as inferred from the following proposition. In practice, we typically employ smooth virtual feedback in early iterations when policies tend to perform worse.

Proposition 5.8. *Let ω be the average supervised training error from $\mathcal{F}$, i.e. $\omega = \min_{f\in\mathcal{F}} \mathbb{E}_{x\sim\mathcal{X}}[\|f([x,0]) - a^*\|]$. Let the rolled-out trajectory of current policy π be $\{a_t\}$. If the average gap between π and π^* is such that $\mathbb{E}_{t\sim Uniform[1:T]}[\|a_t^* - a_{t-1}\|] \geqslant 3\omega + \eta(1+\lambda)$, then using $\{a_t^*\}$ as feedback will cause the trained policy $\hat{\pi}$ to be non-smooth, i.e.:*

$$\mathbb{E}_{t\sim Uniform[1:T]}[\|\widehat{a}_t - \widehat{a}_{t-1}\|] \geqslant \eta, \tag{7}$$

for $\{\widehat{a}_t\}$ the rolled-out trajectory of $\hat{\pi}$.



(a) SIMILE Test Set 1 (b) Supervised Test Set 1

(c) SIMILE Test Set 2 (d) Supervised Test Set 2

Figure 2. Expert (blue) and predicted (red) camera pan angles. Left: SIMILE with <10 iterations. Right: non-smooth policy.

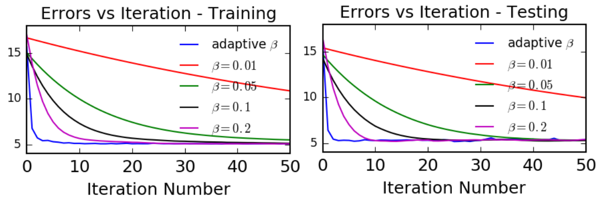


Figure 3. Adaptive versus fixed interpolation parameter β.

6. Experiments

Automated Camera Planning. We evaluate SIMILE in a case study of automated camera planning for sport broadcasting (Chen & Carr, 2015; Chen et al., 2016). Given noisy tracking of players as raw input data $\{x_t\}_{t=1}^T$, and demonstrated pan camera angles from professional human operator as $\{a_t^*\}_{t=1}^T$, the goal is to learn a policy π that produces trajectory $\{a_t\}_{t=1}^T$ that is both smooth and accurate relative to $\{a_t^*\}_{t=1}^T$. Smoothness is critical in camera control: fluid movements which maintain adequate framing are preferable to jittery motions which constantly pursue perfect tracking (Gaddam et al., 2015). In this setting, time horizon T is the duration of the event multiplied by rate of sampling. Thus T tends to be very large.

Smooth Policy Class. We use a smooth policy class following Example 2.2: regression tree ensembles $\mathcal{F}$ regularized by a class of linear autoregressor functions $\mathcal{H}$ (Chen et al., 2016). See Appendix B for more details.

Summary of Results.

- Using our smooth policy class leads to dramatically smoother trajectories than not regularizing using $\mathcal{H}$.
- Using our adaptive learning rate leads to much faster convergence compared to conservative learning rates from SEARN (Daumé III et al., 2009).
- Using smooth feedback ensures stable learning of smooth policies at each iteration.
- Deterministic policy interpolation performs better than stochastic interpolation used in SEARN.

Smooth versus Non-Smooth Policy Classes. Figure 2 shows a comparison of using a smooth policy class versus a non-smooth one (e.g., not using $\mathcal{H}$). We see that our

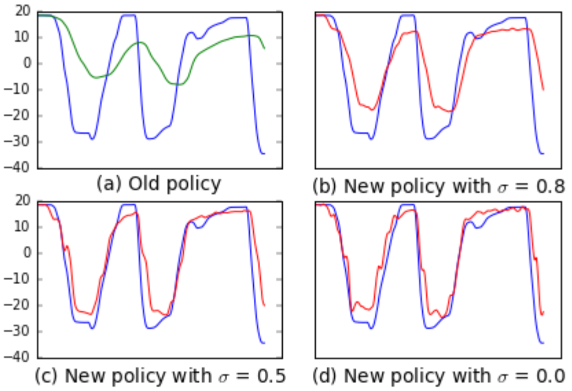


Figure 4. Comparing different values of σ.

approach can reliably learn to predict trajectories that are both smooth and accurate.

Adaptive vs. Fixed β: One can, in principle, train using SEARN, which requires a very conservative β in order to guarantee convergence. In contrast, SIMILE adaptively selects β based on relative empirical loss of π and $\hat{\pi}$ (Line 9 of Algorithm 1). Let $\texttt{error}(\hat{\pi})$ and $\texttt{error}(\pi)$ denote the mean-squared errors of rolled-out trajectories $\{\hat{a}_t\}$, $\{a_t\}$, respectively, w.r.t. ground truth $\{a_t^*\}$. We can set β as:

$$\hat{\beta} = \frac{\texttt{error}(\pi)}{\texttt{error}(\hat{\pi}) + \texttt{error}(\pi)}, \tag{8}$$

which encourages the learner to disregard bad policies when interpolating, thus allowing fast convergence to a good policy (see Theorem 5.6). Figure 3 compares the convergence rate of SIMILE using adaptive β versus conservative fixed values of β commonly used in SEARN (Daumé III et al., 2009). We see that adaptively choosing β enjoys substantially faster convergence. Note that very large fixed β may overshoot and worsen the combined policy after a few initial improvements.

Smooth Feedback Generation: We set the target labels to $\hat{a}_t^n = \sigma a_t^n + (1-\sigma)a_t^*$ for $0 < \sigma < 1$ (Line 6 of Algorithm 1). Larger σ corresponds to smoother ($\hat{a}_t^n$ is closer to a_{t-1}^n) but less accurate target (further from a_t^*), as seen in Figure 4. Figure 5 shows the trade-off between smoothness loss (blue line, measured by first order difference in Proposition 5.8) and imitation loss (red line, measured by mean squared distance) for varying σ. We navigate this trade-off by setting σ closer to 1 in early iterations, and have $\sigma \rightarrow 0$ as n increases. This "gradual increase" produces more stable policies, especially during early iterations where the learning policy tends to perform poorly (as formalized in Proposition 5.8). In Figure 4, when the initial policy (green trajectory) has poor performance, setting smooth targets (Figure 4b) allows learning a smooth policy in the subsequent round, in contrast to more accurate but less stable performance of "difficult" targets with low σ (Figure 4c-d). Figure 6 visualizes the behavior of the the intermediate policies learned by SIMILE, where we can see that each intermediate policy is a smooth policy.

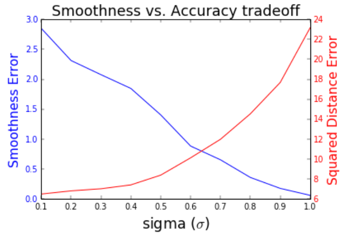


Figure 5.

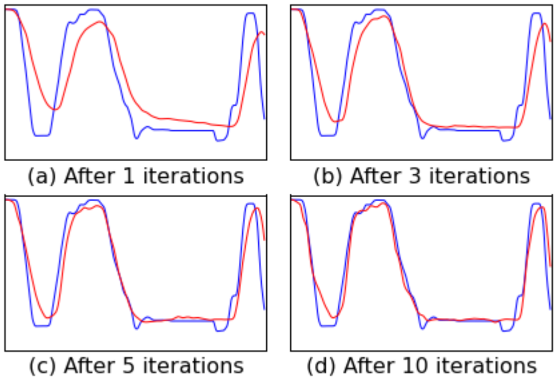


Figure 6. Performance after different number of iterations.

Deterministic vs. Stochastic Interpolation: Finally, we evaluate the benefits of using deterministic policy averaging intead of stochastically combine different policies, as done in SEARN. To control for other factors, we set β to a fixed value of 0.5, and keep the new training dataset $\mathcal{D}_n$ the same for each iteration n. The average imitation loss of stochastic policy sampling are evaluated after 50 stochastic roll-outs at each iterations. This average stochastic policy error tends to be higher compared to the empirical error of the deterministic trajectory, as seen from Figure 7, and confirms our finding from Corollary 5.3.

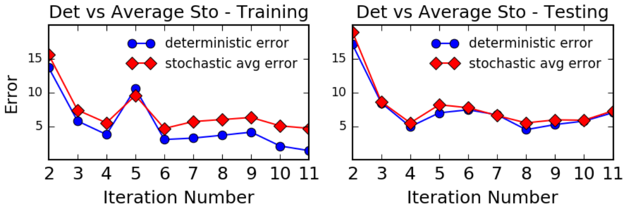


Figure 7. Deterministic policy error vs. average stochastic policy error for $\beta = 0.5$ and 50 roll-outs of the stochastic policies.

7. Conclusion

We formalized the problem of smooth imitation learning for online sequence prediction, which is a variant of imitation learning that uses a notion of a smooth policy class. We proposed SIMILE (**S**mooth **IMI**tation **LE**arning), which is an iterative learning reduction approach to learning smooth policies from expert demonstrations in a continuous and dynamic environment. SIMILE utilizes an adaptive learning rate that provably allows much faster convergence compared to previous learning reduction approaches, and also enjoys better sample complexity than previous work by being fully deterministic and allowing for virtual simulation of training labels. We validated the efficiency and practicality of our approach on a setting of online camera planning.

References

Abbeel, Pieter and Ng, Andrew Y. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2004.

Abbeel, Pieter and Ng, Andrew Y. Exploration and apprenticeship learning in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2005.

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

Bagnell, J Andrew, Kakade, Sham M, Schneider, Jeff G, and Ng, Andrew Y. Policy search by dynamic programming. In *Neural Information Processing Systems (NIPS)*, 2003.

Caruana, Rich and Niculescu-Mizil, Alexandru. An empirical comparison of supervised learning algorithms. In *International Conference on Machine Learning (ICML)*, 2006.

Chen, Jianhui and Carr, Peter. Mimicking human camera operators. In *IEEE Winter Conference Applications of Computer Vision (WACV)*, 2015.

Chen, Jianhui, Le, Hoang M., Carr, Peter, Yue, Yisong, and Little, James J. Learning online smooth predictors for real-time camera planning using recurrent decision trees. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Criminisi, Antonio, Shotton, Jamie, and Konukoglu, Ender. 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.

Daumé III, Hal, Langford, John, and Marcu, Daniel. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

Gaddam, Vamsidhar Reddy, Eg, Ragnhild, Langseth, Ragnar, Griwodz, Carsten, and Halvorsen, Pål. The cameraman operating my virtual camera is artificial: Can the machine be as good as a human&quest. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 11(4):56, 2015.

He, He, Eisner, Jason, and Daume, Hal. Imitation learning by coaching. In *Neural Information Processing Systems (NIPS)*, 2012.

Jain, Ashesh, Wojcik, Brian, Joachims, Thorsten, and Saxena, Ashutosh. Learning trajectory preferences for manipulators via iterative improvement. In *Neural Information Processing Systems (NIPS)*, 2013.

Kakade, Sham and Langford, John. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2002.

Lagoudakis, Michail and Parr, Ronald. Reinforcement learning as classification: Leveraging modern classifiers. In *International Conference on Machine Learning (ICML)*, 2003.

Langford, John and Zadrozny, Bianca. Relating reinforcement learning performance to classification performance. In *International Conference on Machine Learning (ICML)*, 2005.

Mason, Llew, Baxter, Jonathan, Bartlett, Peter L, and Frean, Marcus. Functional gradient techniques for combining hypotheses. In *Neural Information Processing Systems (NIPS)*, 1999.

Ratliff, Nathan, Silver, David, and Bagnell, J. Andrew. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, 2009.

Ross, Stéphane and Bagnell, Drew. Efficient reductions for imitation learning. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Ross, Stephane, Gordon, Geoff, and Bagnell, J. Andrew. A reduction of imitation learning and structured prediction to no-regret online learning. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Srebro, Nathan, Sridharan, Karthik, and Tewari, Ambuj. Smoothness, low noise and fast rates. In *Neural Information Processing Systems (NIPS)*, 2010.

Syed, Umar and Schapire, Robert E. A reduction from apprenticeship learning to classification. In *Neural Information Processing Systems (NIPS)*, 2010.

Wold, Herman. A study in the analysis of stationary time series, 1939.

Supplementary Materials for Smooth Imitation Learning for Online Sequence Prediction

A. Detailed Theoretical Analysis and Proofs

A.1. Proof of lemma 5.1

Lemma Statement. *(Lemma 5.1) For a fixed x, define $\pi([x,a]) \triangleq \varphi(a)$. If φ is non-negative and H-smooth w.r.t. a., then:*

$$\forall a, a' : \left(\varphi(a) - \varphi(a')\right)^2 \leqslant 6H\left(\varphi(a) + \varphi(a')\right)\|a - a'\|^2 .$$

The proof of Lemma 5.1 rests on 2 properties of H-smooth functions (differentiable) in $\mathbb{R}^1$, as stated below

Lemma A.1 (Self-bounding property of Lipschitz-smooth functions)**.** *Let $\phi : \mathbb{R} \to \mathbb{R}$ be an H-smooth non-negative function. Then for all $a \in \mathbb{R}$: $|\nabla\phi(a)| \leqslant \sqrt{4H\phi(a)}$*

Proof. By mean value theorem, for any a, a' we have $\exists\ \eta \in (a, a')$ (or (a', a)) such that $\phi(a') = \phi(a) + \nabla\phi(\eta)(a' - a)$. Since ϕ is non-negative,

$$\begin{aligned} 0 \leqslant \phi(a') &= \phi(a) + \nabla\phi(a)(a' - a) \\ &\quad + (\nabla\phi(\eta) - \nabla\phi(a))(a' - a) \\ &\leqslant \phi(a) + \nabla\phi(a)(a' - a) + H|\eta - a||a' - a| \\ &\leqslant \phi(a) + \nabla\phi(a)(a' - a) + H|a' - a|^2 \end{aligned}$$

Choosing $a' = a - \frac{\nabla\phi(a)}{2H}$ proves the lemma. □

Lemma A.2 (1-d Case (Srebro et al., 2010))**.** *Let $\phi : \mathbb{R} \to \mathbb{R}$ be an H-smooth non-negative function. Then for all $a, a' \in \mathbb{R}$:*

$$\left(\phi(a) - \phi(a')\right)^2 \leqslant 6H\left(\phi(a) + \phi(a')\right)\left(a - a'\right)^2$$

Proof. As before, $\exists\eta \in (a, a')$ such that $\phi(a') - \phi(a) = \nabla\phi(\eta)(a' - a)$. By assumption of ϕ, we have $|\nabla\phi(\eta) - \nabla\phi(a)| \leqslant H|\eta - a| \leqslant H|a' - a|$. Thus we have:

$$|\nabla\phi(\eta)| \leqslant |\nabla\phi(a)| + H|a - a'| \tag{9}$$

Consider two cases:

Case 1: If $|a - a'| \leqslant \frac{|\nabla\phi(a)|}{5H}$, then by equation 9 we have $|\nabla\phi(\eta)| \leqslant 6/5|\nabla\phi(a)|$. Thus

$$\begin{aligned} \left(\phi(a) - \phi(a')\right)^2 &= (\nabla\phi(\eta))^2\left(a - a'\right)^2 \\ &\leqslant \frac{36}{25}(\nabla\phi(a))^2\left(a - a'\right)^2 \\ &\leqslant \frac{144}{25}H\phi(a)\left(a - a'\right)^2 \end{aligned}$$

by lemma A.1. Therefore, $\left(\phi(a) - \phi(a')\right)^2 \leqslant 6H\phi(a)\left(a - a'\right)^2 \leqslant 6H\left(\phi(a) + \phi(a')\right)\left(a - a'\right)^2$

Case 2: If $|a - a'| > \frac{|\nabla\phi(a)|}{5H}$, then equation 9 gives $|\nabla\phi(\eta)| \leqslant 6H|a - a'|$. Once again

$$\begin{aligned} \left(\phi(a) - \phi(a')\right)^2 &= \left(\phi(a) - \phi(a')\right)\nabla\phi(\eta)\left(a - a'\right) \\ &\leqslant |\left(\phi(a) - \phi(a')\right)||\nabla\phi(\eta)||\left(a - a'\right)| \\ &\leqslant |\left(\phi(a) - \phi(a')\right)|\left(6H\left(a - a'\right)^2\right) \\ &\leqslant 6H\left(\phi(a) + \phi(a')\right)\left(a - a'\right)^2 \end{aligned}$$

□

Proof of Lemma 5.1. The extension to the multi-dimensional case is straightforward. For any $a, a' \in \mathbb{R}^k$, consider the function $\phi : \mathbb{R} \to \mathbb{R}$ such that $\phi(t) = \varphi((1-t)a + ta')$, then ϕ is a differentiable, non-negative function and $\nabla_t(\phi(t)) = \langle\nabla\varphi(a + t(a' - a)), a' - a\rangle$. Thus:

$$\begin{aligned} &|\phi'(t_1) - \phi'(t_2)| = |\langle\nabla\varphi(a + t_1(a' - a)) - \\ &\qquad\qquad \nabla\varphi(a + t_2(a' - a)), a' - a\rangle| \\ &\leqslant \left\|\nabla\varphi(a + t_1(a' - a)) - \nabla\varphi(a + t_2(a' - a))\right\|_* \|a' - a\| \\ &\leqslant H|t_1 - t_2|\left\|a - a'\right\|^2 \end{aligned}$$

Therefore ϕ is an $H\left\|a - a'\right\|^2$-smooth function in $\mathbb{R}$. Apply lemma A.2 to ϕ, we have:

$$(\phi(1) - \phi(0))^2 \leqslant 6H\left\|a - a'\right\|^2(\phi(1) + \phi(0))(1 - 0)^2$$

which is the same as $(\varphi(a) - \varphi(a'))^2 \leqslant 6H(\varphi(a) + \varphi(a'))\|a - a'\|^2$ □

A.2. Proof of lemma 5.2

Lemma Statement. *(Lemma 5.2) Given any starting state s_0, sequentially execute π_{det} and π_{sto} to obtain two separate trajectories $\boldsymbol{A} = \{a_t\}_{t=1}^T$ and $\tilde{\boldsymbol{A}} = \{\tilde{a}_t\}_{t=1}^T$ such that $a_t = \pi_{det}(s_t)$ and $\tilde{a}_t = \pi_{sto}(\tilde{s}_t)$, where $s_t = [x_t, a_{t-1}]$ and $\tilde{s}_t = [x_t, \tilde{a}_{t-1}]$. Assuming the policies are stable as per Condition 1, we have $\mathbb{E}_{\tilde{\mathbf{A}}}[\tilde{a}_t] = a_t\ \forall t = 1, \ldots, T$, where the expectation is taken over all random roll-outs of π_{sto}.*

Proof. Given a starting state s_0, we prove by induction that $\mathbb{E}_{\tilde{\mathbf{A}}}[\tilde{a}_t] = a_t$.

It is easily seen that the claim is true for $t = 1$.

Now assuming that $\mathbb{E}_{\tilde{\mathbf{A}}}[\tilde{a}_{t-1}] = a_{t-1}$. We have

$$\begin{aligned}\mathbb{E}_{\tilde{\mathbf{A}}}[\tilde{a}_t] &= \mathbb{E}_{\tilde{\mathbf{A}}}[\mathbb{E}[\tilde{a}_t|\tilde{s}_t]]\\ &= \mathbb{E}_{\tilde{\mathbf{A}}}[\beta\hat{\pi}(\tilde{s}_t) + (1-\beta)\pi(\tilde{s}_t)]\\ &= \beta\mathbb{E}_{\tilde{\mathbf{A}}}[\hat{\pi}(\tilde{s}_t)] + (1-\beta)\mathbb{E}_{\tilde{\mathbf{A}}}[\pi(\tilde{s}_t)]\end{aligned}$$

Thus:

$$\begin{aligned}\left\|\mathbb{E}_{\tilde{\mathbf{A}}}[\tilde{a}_t] - a_t\right\| &= \left\|\mathbb{E}_{\tilde{\mathbf{A}}}[\tilde{a}_t] - \beta\hat{\pi}(s_t) - (1-\beta)\pi(s_t)\right\|\\ &= \|\beta\mathbb{E}_{\tilde{\mathbf{A}}}[\hat{\pi}(\tilde{s}_t)] + (1-\beta)\mathbb{E}_{\tilde{\mathbf{A}}}[\pi(\tilde{s}_t)]\\ &\qquad - \beta\hat{\pi}(s_t) - (1-\beta)\pi(s_t)\|\\ &\leqslant \beta\left\|\mathbb{E}_{\tilde{\mathbf{A}}}[\hat{\pi}(\tilde{s}_t)] - \hat{\pi}(s_t)\right\|\\ &\quad + (1-\beta)\left\|\mathbb{E}_{\tilde{\mathbf{A}}}[\pi(\tilde{s}_t)] - \pi(s_t)\right\|\\ &\leqslant \beta\left\|\mathbb{E}_{\tilde{\mathbf{A}}}[\tilde{a}_{t-1}] - a_{t-1}\right\|\\ &\quad + (1-\beta)\left\|\mathbb{E}_{\tilde{\mathbf{A}}}[\tilde{a}_{t-1}] - a_{t-1}\right\|\\ &= 0\end{aligned}$$

per inductive hypothesis. Therefore we conclude that $\mathbb{E}_{\tilde{\mathbf{A}}}[\tilde{a}_t] = a_t \;\; \forall t = 1, \ldots, T$ □

A.3. Proof of theorem 5.6 and corollary 5.7 - Main policy improvement results

In this section, we provide the proof to theorem 5.6 and corollary 5.7.

Theorem Statement. *(theorem 5.6) Assume ℓ is convex and L-Lipschitz-continuous, and Condition 2 holds. Let $\epsilon = \max\limits_{s\sim d_\pi}\|\hat{\pi}(s) - \pi(s)\|$. Then for $\beta \in (0,1)$:*

$$\ell_{\pi'}(\pi') - \ell_\pi(\pi) \leqslant \frac{\beta\gamma\epsilon L}{(1-\beta)(1-\gamma)} + \beta(\ell_\pi(\hat{\pi}) - \ell_\pi(\pi)).$$

Proof. First let's review the notations: 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 $s_0 \xrightarrow{\pi} s_1 \xrightarrow{\pi} \ldots \xrightarrow{\pi} s_T$, where $s_t = [x_t, \pi(s_{t-1})]$, with x_t encodes the featurized input at current time t, and $\pi(s_{t-1})$ encodes the dependency on previous predictions. Let $\ell(\pi(s))$ be the loss of taking action $\pi(s)$ at state s, we can define the trajectory loss of policy π from starting state s_0 as

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

For a starting state distribution μ, we define policy loss of π as the expected loss along trajectories induced by π: $\ell_\pi(\pi) = \mathbb{E}_{s_0\sim\mu}[\ell(\pi|s_0)]$. Policy loss $\ell_\pi(\pi)$ can be understood as

$$\ell_\pi(\pi) = \int_{s_0\sim\mu}\mathop{\mathbb{E}}_{x_t\sim\mathcal{X}}\frac{1}{T}\left[\sum_{t=1}^{T}\ell(\pi(s_t))\right]d\mu(s_0)$$

To prove policy improvement, we skip the subscript of algorithm 1 to consider general policy update rule within each iteration:

$$\pi' = \pi_{new} = \beta\hat{\pi} + (1-\beta)\pi \qquad (10)$$

where $\pi = \pi_{old}$ is the current policy (combined up until the previous iteration), $\hat{\pi}$ is the trained model from calling the base regression routine $\texttt{Train}(\mathbf{S}, \widehat{\mathbf{A}}|h)$. Learning rate (step-size) β may be adaptively chosen in each iteration. Recall that this update rule reflects deterministic interpolation of two policies.

We are interested in quantifying the policy improvement when updating π to π'. Specifically, we want to bound

$$\Gamma = \ell_{\pi'}(\pi') - \ell_\pi(\pi)$$

where $\ell_\pi(\pi)$ (respectively $\ell_{\pi'}(\pi')$) denotes the trajectory loss of π (respectively π') on the state distribution induced by π (resp. π')

We will bound the loss difference of old and new policies conditioned on a common starting state s_0. Based on update rule (10), consider rolling out π' and π from the same starting state s_0 to obtain two separate sequences $\pi' \longmapsto \{s_0 \to s'_1 \ldots \to s'_T\}$ and $\pi \longmapsto \{s_0 \to s_1 \ldots \to s_T\}$ corresponding to the same stream of inputs $x_1, \ldots, x_T$.

$$\begin{aligned}\Gamma(s_0) &= \frac{1}{T}\sum_{t=1}^{T}\ell(\pi'(s'_t)) - \ell(\pi(s_t))\\ &= \frac{1}{T}\sum_{t=1}^{T}\ell(\pi'(s'_t)) - \ell(\pi'(s_t)) + \ell(\pi'(s_t)) - \ell(\pi(s_t))\end{aligned} \qquad (11)$$

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

$$\begin{aligned}\ell(\pi'(s_t)) &= \ell(\beta\hat{\pi}(s_t) + (1-\beta)\pi(s_t))\\ &\leqslant \beta\ell(\hat{\pi}(s_t)) + (1-\beta)\ell(\pi(s_t))\end{aligned}$$

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

$$\begin{aligned}\ell(\pi'(s'_t)) - \ell(\pi(s_t)) &\leqslant \ell(\pi'(s'_t))) - \ell(\pi'(s_t))\\ &\quad + \beta\left[\ell(\hat{\pi}(s_t)) - \ell(\pi(s_t))\right]\end{aligned}$$

Since ℓ is L-Lipschitz continuous, we have

$$\begin{aligned}\ell(\pi'(s'_t)) - \ell(\pi'(s_t)) &\leqslant L\left\|\pi'(s'_t) - \pi'(s_t)\right\|\\ &\leqslant L\gamma\left\|s'_t - s_t\right\| \qquad (12)\end{aligned}$$

where (12) is due to the smoothness condition [2] of policy class Π. Given a policy class Π with $\gamma < 1$, the following claim can be proved by induction:
<u>Claim:</u> $\|s'_t - s_t\| \leqslant \frac{\beta\epsilon}{(1-\beta)(1-\gamma)}$

Proof. For the base case, given the same starting state s_0, we have $s'_1 = [x_1, \pi'(s_0)]$ and $s_1 = [x_1, \pi(s_0)]$. Thus $\|s'_1 - s_1\| = \|\pi'(s_0) - \pi(s_0)\| = \|\beta\hat{\pi}(s_0) + (1-\beta)\pi(s_0) - \pi(s_0)\| = \beta\|\hat{\pi}(s_0) - \pi(s_0)\| \leqslant \beta\epsilon \leqslant \frac{\beta\epsilon}{(1-\beta)(1-\gamma)}$.

In the inductive case, assume we have $\|s'_{t-1} - s_{t-1}\| \leqslant$

$\frac{\beta\epsilon}{(1-\beta)(1-\gamma)}$. Then similar to before, the definition of s'_t and s_t leads to

$$\begin{aligned}\left\|s'_t - s_t\right\| &= \left\|\left[x_t, \pi'(s'_{t-1})\right] - \left[x_t, \pi(s_{t-1})\right]\right\| \\ &= \left\|\pi'(s'_{t-1}) - \pi(s_{t-1})\right\| \\ &\leqslant \left\|\pi'(s'_{t-1}) - \pi'(s_{t-1})\right\| + \left\|\pi'(s_{t-1}) - \pi(s_{t-1})\right\| \\ &\leqslant \gamma \left\|s'_{t-1} - s_{t-1}\right\| + \beta \left\|\hat{\pi}(s_{t-1}) - \pi(s_{t-1})\right\| \\ &\leqslant \gamma \frac{\beta\epsilon}{(1-\beta)(1-\gamma)} + \beta\epsilon \\ &\leqslant \frac{\beta\epsilon}{(1-\beta)(1-\gamma)}\end{aligned}$$

□

Applying the claim to equation (12), we have

$$\ell(\pi'(s'_t)) - \ell(\pi'(s_t)) \leqslant \frac{\beta\gamma\epsilon L}{(1-\beta)(1-\gamma)}$$

which leads to

$$\begin{aligned}\ell(\pi'(s'_t) - \ell(\pi(s_t))) \leqslant & \frac{\beta\gamma\epsilon L}{(1-\beta)(1-\gamma)} \\ & + \beta(\ell(\hat{\pi}(s_t)) - \ell(\pi(s_t))) \quad (13)\end{aligned}$$

Integrating (13) over the starting state $s_0 \sim \mu$ and input trajectories $\{x_t\}_{t=1}^T$, we arrive at the policy improvement bound:

$$\ell_{\pi'}(\pi') - \ell_\pi(\pi) \leqslant \frac{\beta\gamma\epsilon L}{(1-\beta)(1-\gamma)} + \beta(\ell_\pi(\hat{\pi}) - \ell_\pi(\pi))$$

where $\ell_\pi(\hat{\pi})$ is the expected loss of the trained policy $\hat{\pi}$ on the state distribution induced by policy π (reduction term, analogous to policy advantage in the traditional MDP terminologies (Kakade & Langford, 2002)) □

This means in the worst case, as we choose $\beta \to 0$, we have $[\ell_{\pi'}(\pi') - \ell_\pi(\pi)] \to 0$, meaning the new policy does not degrade much for a small choice of β. However if $\ell_\pi(\hat{\pi}) - \ell_\pi(\pi) \ll 0$, we can choose β to enforce monotonic improvement of the policy by adaptively choosing β that minimizes the right-hand side. In particular, let the reduction term be $\Delta = \ell_\pi(\pi) - \ell_\pi(\hat{\pi}) > 0$ and let $\delta = \frac{\gamma\epsilon L}{1-\gamma}$, then for $\beta = \frac{\Delta-\delta}{2\Delta}$we have the following monotonic policy improvement:

$$\ell_{\pi'}(\pi') - \ell_\pi(\pi) \leqslant -\frac{(\Delta-\delta)^2}{2(\Delta+\delta)}$$

A.4. Proof of theorem 5.5 - T-dependent improvement

Theorem Statement. *(theorem 5.5) Assume ℓ is convex and L-Lipschitz, and Condition 1 holds. Let $\epsilon = \max_{s\sim d_\pi} \|\hat{\pi}(s) - \pi(s)\|$. Then:*

$$\ell_{\pi'}(\pi') - \ell_\pi(\pi) \leqslant \beta\epsilon LT + \beta\left(\ell_\pi(\hat{\pi}) - \ell_\pi(\pi)\right).$$

In particular, choosing $\beta \in (0, 1/T)$ yields:

$$\ell_{\pi'}(\pi') - \ell_\pi(\pi) \leqslant \epsilon L + \beta\left(\ell_\pi(\hat{\pi}) - \ell_\pi(\pi)\right).$$

Proof. The proof of theorem 5.5 largely follows the structute of theorem 5.6, except that we are using the slighty weaker Condition 1 which leads to weaker bound on the policy improvement that depends on the trajectory horizon T. For any state s_0 taken from the starting state distribution μ, sequentially roll-out policies π' and π to receive two separate trajectories $\pi' : s_0 \to s'_1 \to \ldots \to s'_T$ and $\pi' : s_0 \to s_1 \to \ldots \to s_T$. Consider a pair of states $s'_t = [x_t, \pi'(s'_{t-1})]$ and $s_t = [x_t, \pi(s_{t-1})]$ corresponding to the same input feature x_t, as before we can decompose $\ell(\pi'(s'_t)) - \ell(\pi(s_t)) = \ell(\pi'(s'_t)) - \ell(\pi'(s_t)) + \ell(\pi'(s_t)) - \ell(\pi(s_t)) \leqslant L\left\|\pi'(s'_t) - \pi'(s_t)\right\| + \beta(\ell(\hat{\pi}(s_t)) - \ell(\pi(s_t)))$ due to convexity and L-Lipschitz continuity of ℓ.

Condition 1 further yields: $\ell(\pi'(s'_t)) - \ell(\pi(s_t)) \leqslant L\left\|s'_t - s_t\right\| + \beta(\ell(\hat{\pi}(s_t)) - \ell(\pi(s_t)))$. By the construction of the states, note that

$$\begin{aligned}\left\|s'_t - s_t\right\| &= \left\|\pi'(s'_{t-1}) - \pi(s_{t-1})\right\| \\ &\leqslant \left\|\pi'(s'_{t-1}) - \pi'(s_{t-1})\right\| + \left\|\pi'(s_{t-1}) - \pi(s_{t-1})\right\| \\ &\leqslant \left\|s'_{t-1} - s_{t-1}\right\| + \beta(\left\|\hat{\pi}(s_{t-1}) - \pi(s_{t-1})\right\|) \\ &\leqslant \left\|s'_{t-1} - s_{t-1}\right\| + \beta\epsilon\end{aligned}$$

(by condition 1 and definition of ϵ).

From here, one can use this recursive relation to easily show that $\left\|s'_t - s_t\right\| \leqslant \beta\epsilon t$ for all $t \in [1, T]$.

Averaging over the T time steps and integrating over the starting state distribution, we have:

$$\begin{aligned}\ell_{\pi'}(\pi') - \ell_\pi(\pi) &\leqslant \beta\epsilon L(T+1)/2 + \beta(\ell_\pi(\hat{\pi}) - \ell_\pi(\pi)) \\ &\leqslant \beta\epsilon LT + \beta(\ell_\pi(\hat{\pi}) - \ell_\pi(\pi))\end{aligned}$$

In particular, $\beta \in (0, 1/T)$ yields $\ell_{\pi'}(\pi') - \ell_\pi(\pi) \leqslant \epsilon L + \beta(\ell_\pi(\hat{\pi}) - \ell_\pi(\pi))$. □

A.5. Proof of proposition 5.8 - smooth expert proposition

Proposition Statement. *(Proposition 5.8) Let ω be the average supervised training error from $\mathcal{F}$, i.e. $\omega = \min_{f\in\mathcal{F}} \mathbb{E}_{x\sim\mathcal{X}}\left[\|f([x,0]) - a^*\|\right]$. Let the rolled-out trajectory of current policy π be $\{a_t\}$. If the average gap between π and π^* is such that $\mathbb{E}_{t\sim Uniform[1:T]}\left[\|a^*_t - a_{t-1}\|\right] \geqslant 3\omega + \eta(1+\lambda)$, then using $\{a^*_t\}$ as feedback will cause the trained policy $\hat{\pi}$ to be non-smooth, i.e.:*

$$\mathbb{E}_{t\sim Uniform[1:T]}\left[\|\hat{a}_t - \hat{a}_{t-1}\|\right] \geqslant \eta,$$

for $\{\hat{a}_t\}$ the rolled-out trajectory of $\hat{\pi}$.

Proof. Recall that Π_λ is formed by regularizing a class of supervised learners $\mathcal{F}$ with the singleton class of smooth function $\mathcal{H} \triangleq \{h(a) = a\}$, via a hyper-parameter λ

that controls the trade-off between being close to the two classes.

Minimizing over Π_λ can be seen as a regularized optimization problem:

$$\begin{aligned}\hat{\pi}(x,a) &= \operatorname*{argmin}_{\pi\in\Pi} \ell(\pi([x,a])) \\ &= \operatorname*{argmin}_{f\in\mathcal{F},h\in\mathcal{H}} (f(x,a)-a^*)^2 + \lambda(f(x,a)-h(a))^2 \\ &= \operatorname*{argmin}_{f\in\mathcal{F}}(f(x,a)-a^*)^2 + \lambda(f(x,a)-a)^2\end{aligned} \tag{14}$$

where hyper-parameter λ trades-off the distance of $f(x,a)$ relative to a (smoothness) and a^* (imitation accuracy), and $a \in \mathbb{R}^1$.

Such a policy π, at execution time, corresponds to the regularized minimizer of:

$$\begin{aligned}a_t &= \pi([x,a_{t-1}]) \\ &= \operatorname*{argmin}_{a} \|a - f([x_t,a_{t-1}])\|^2 + \lambda\|a-a_{t-1}\|^2 \\ &= \frac{f([x_t,a_{t-1}]) + \lambda a_{t-1}}{1+\lambda}\end{aligned} \tag{15}$$

where $f \in \mathcal{F}$ is the minimizer of equation 14

Thus we enforce smoothness of learning policy from Π_λ by encouraging low first order difference of consecutive actions of the executed trajectory $\{a_t\}$. In practice, we may contrain this first order difference relative to the human trajectory $\frac{1}{T}\sum_{t=1}^{T}\|a_t - a_{t-1}\| \leqslant \eta$, where $\eta \propto \frac{1}{T}\sum_{t=1}^{T}\|a_t^* - a_{t-1}^*\|$.

Consider any given iteration with the following set-up: we execute old policy $\pi = \pi_{old}$ to get rolled-out trajectory $\{a_t\}_{t=1}^T$. Form the new data set as $\mathcal{D} = \{(s_t, a_t^*)\}_{t=1}^T$ with predictors $s_t = [x_t, a_{t-1}]$ and feedback labels simply the human actions a_t^*. Use this data set to train a policy $\hat{\pi}$ by learning a supervised $\hat{f} \in \mathcal{F}$ from $\mathcal{D}$. Similar to π, the execution of $\hat{\pi}$ corresponds to $\hat{a}_t$ where:

$$\begin{aligned}\hat{a}_t &= \hat{\pi}([x_t,\hat{a}_{t-1}]) \\ &= \operatorname*{argmin}_{a} \left\|a - \hat{f}([x_t,\hat{a}_{t-1}])\right\|^2 + \lambda\|a-\hat{a}_{t-1}\|^2 \\ &= \frac{\hat{f}([x_t,\hat{a}_{t-1}]) + \lambda \hat{a}_{t-1}}{1+\lambda}\end{aligned} \tag{16}$$

Denote by f_0 the "naive" supervised learner from $\mathcal{F}$. In other words, $f_0 = \operatorname*{argmin}_{f\in\mathcal{F}} \sum\limits_{t=1}^{T}\|f([x_t,0]) - a_t^*\|^2$. Let ω be the average gap between human trajectory and the rolled-out trajectory of f_0, i.e.

$$\omega = \frac{1}{T}\sum_{t=1}^{T}\|f_0([x_t,0]) - a_t^*\|$$

Note that it is reasonable to assume that the average errors of f and $\hat{f}$ are no worse than f_0, since in the worst case we can simply discard the extra features a_{t-1} (resp. $\hat{a}_{t-1}$) of f (resp. $\hat{f}$) to recover the performance of the naive learner f_0:

$$\frac{1}{T}\sum_{t=1}^{T}\|f([x_t,a_{t-1}]) - a_t^*\| \leqslant \omega$$

$$\frac{1}{T}\sum_{t=1}^{T}\left\|\hat{f}([x_t,\hat{a}_{t-1}]) - a_t^*\right\| \leqslant \omega$$

Assume that the old policy $\pi = \pi_{old}$ is "bad" in the sense that the rolled-out trajectory $\{a_t\}_{t=1}^T$ differs substantially from human trajectory $\{a_t^*\}_{t=1}^T$. Specifically, denote the gap:

$$\frac{1}{T}\sum_{t=1}^{T}\|a_t^* - a_{t-1}\| = \Omega \gg \omega$$

This means the feedback correction a_t^* to $s_t = [x_t, a_{t-1}]$ is not smooth. We will show that the trained policy $\hat{\pi}$ from $\mathcal{D}$ will not be smooth.

From the definition of a_t and $\hat{a}_t$ from equations 15 and 16, we have for each t:

$$a_t - \hat{a}_t = \frac{\lambda}{1+\lambda}(a_{t-1} - \hat{a}_{t-1}) + \frac{f([x_t,a_{t-1}]) - \hat{f}([x_t,\hat{a}_{t-1}])}{1+\lambda}$$

Applying triangle inequality and summing up over t, we have:

$$\frac{1}{T}\sum_{t=1}^{T}\|a_t - \hat{a}_t\| \leqslant 2\omega$$

From here we can provide a lower bound on the smoothness of the new trajectory $\hat{a}_t$, as defined by the first order difference $\frac{1}{T}\sum_{t=1}^{T}\|\hat{a}_t - \hat{a}_{t-1}\|$. By definition of $\hat{a}_t$:

$$\begin{aligned}\|\hat{a}_t - \hat{a}_{t-1}\| &= \left\|\frac{\hat{f}([x_t,\hat{a}_{t-1}]) - \hat{a}_{t-1}}{1+\lambda}\right\| \\ &= \left\|\frac{\hat{f}([x_t,\hat{a}_{t-1}]) - a_t^* + a_t^* - a_{t-1} + a_{t-1} - \hat{a}_{t-1}}{1+\lambda}\right\| \\ &\geqslant \frac{\|a_t^* - a_{t-1}\| - \left\|\hat{f}([x_t,\hat{a}_{t-1}]) - a_t^*\right\| - \|a_{t-1} - \hat{a}_{t-1}\|}{1+\lambda}\end{aligned}$$

Again summing up over t and taking the average, we obtain:

$$\frac{1}{T}\sum_{t=1}^{T}\|\hat{a}_t - \hat{a}_{t-1}\| \geqslant \frac{\Omega - 3\omega}{1+\lambda}$$

Hence for $\Omega \gg \omega$, meaning the old trajectory is sufficiently far away from the ideal human trajectory, setting the learning target to be the ideal human actions will cause the learned trajectory to be non-smooth. □

B. Imitation Learning for Online Sequence Prediction With Smooth Regression Forests

B.1. Variant of SIMILE Using Smooth Regression Forest Policy Class

We provide a specific instantiation of algorithm 1 that we used for our experiment, based on a policy class Π as a smooth regularized version of the space of tree-based ensembles. In particular, $\mathcal{F}$ is the space of random forests and $\mathcal{H}$ is the space of linear auto-regressors $\mathcal{H} \triangleq \{h(a_{t-1:t-\tau}) = \sum_{i=1}^{\tau} c_i a_{t-i}\}$. In combination, $\mathcal{F}$ and $\mathcal{H}$ form a complex tree-based predictor that can predict smooth sequential actions.

Empirically, decision tree-based ensembles are among the best performing supervised machine learning method (Caruana & Niculescu-Mizil, 2006; Criminisi et al., 2012). Due to the piece-wise constant nature of decision tree-based prediction, the results are inevitably non-smooth. We propose a recurrent extension based on $\mathcal{H}$, where the prediction at the leaf node is not necessarily a constant, but rather is a smooth function of both static leaf node prediction and its previous predictions. By merging the powerful tree-based policy class with a linear auto-regressor, we provide a novel approach to train complex models that can accommodate smooth sequential prediction using model-based smooth regularizer, at the same time leveraging the expressiveness of complex model-free function class (one can similarly apply the framework to the space of neural networks). Algorithm 2, which is based on SIMILE, describes in more details our training procedure used for the automated camera planning experiment. We first describe the role of the linear autoregressor class $\mathcal{H}$, before discussing how to incorporate $\mathcal{H}$ into decision tree training to make smooth prediction (see the next section).

The autoregresor $h_\pi(a_{-1}, \ldots, a_{-\tau})$ is typically selected from a class of autoregressors $\mathcal{H}$. In our experiments, we use regularized linear autoregressors as $\mathcal{H}$.

Algorithm 2 Imitation Learning for Online Sequence Prediction with Smooth Regression Forest

```
Input: Input features $\mathbf{X} = \{x_t\}_{t=1}^T$, expert demonstration
    $\mathbf{A}^* = \{a_t^*\}_{t=1}^T$, base routine Forest, past horizon $\tau$,
    sequence of $\sigma \in (0,1)$
1: Initialize $\mathbf{A}_0 \leftarrow \mathbf{A}^*, \mathbf{S}_0 \leftarrow \{[x_{t:t-\tau}, a_{t-1:t-\tau}^*]\}$,
       $h_0 = \operatorname*{argmin}_{c_1,\ldots,c_\tau} \sum_{t=1}^T \left(a_t^* - \sum_{i=1}^\tau c_i a_{t-i}^*\right)^2$
2: Initial policy $\pi_0 = \hat{\pi}_0 \leftarrow$Forest$(\mathbf{S}_0, \mathbf{A}_0 |\ h_0)$
3: for $n = 1, \ldots, N$ do
4:     $\mathbf{A}_n = \{a_t^n\} \leftarrow \{\pi_{n-1}([x_{t:t-\tau}, a_{t-1:t-\tau}^{n-1}])\}$
           //sequential roll-out old policy
5:     $\mathbf{S}_n \leftarrow \{s_t^n = [x_{t:t-\tau}, a_{t-1:t-\tau}^n]\}$    //Form states in 1d case
6:     $\widehat{\mathbf{A}}_n = \{\hat{a}_t^n = \sigma a_t^n + (1-\sigma) a_t^*\}\ \forall s_t^n \in \mathbf{S}_n$
           // collect smooth 1-step feedback
7:     $h_n = \operatorname*{argmin}_{c_1,\ldots,c_\tau} \sum_{t=1}^T \left(\hat{a}_t^n - \sum_{i=1}^\tau c_i \hat{a}_{t-i}^n\right)^2$    //update $c_i$
           via regularized least square
8:     $\hat{\pi}_n \leftarrow$Forest$(\mathbf{S}_n, \widehat{\mathbf{A}}_n |\ h_n)$    // train with smooth
           decision forests. See section B.2
9:     $\beta \leftarrow \frac{\texttt{error}(\pi)}{\texttt{error}(\hat{\pi}) + \texttt{error}(\pi)}$    //set $\beta$ to weighted
           empirical errors
10:    $\pi_n = \beta \hat{\pi}_n + (1-\beta)\pi_{n-1}$    // update policy
11: end for
output Last policy $\pi_N$
```

Consider a generic learning policy π with a rolled-out trajectory $\mathbf{A} = \{a_t\}_{t=1}^T$ corresponding to the input sequence $\mathbf{X} = \{x_t\}_{t=1}^T$. We form the state sequence $\mathbf{S} = \{s_t\}_{t=1}^T = \{[x_t, \ldots, x_{t-\tau}, a_{t-1}, \ldots, a_{t-\tau}]\}_{t=1}^T$, where τ indicates the past horizon that is adequate to approximately capture the full state information. We approximate the smoothness of the trajectory $\mathbf{A}$ by a linear autoregressor

$$h_\pi \equiv h_\pi(s_t) \equiv \sum_{i=1}^{\tau} c_i a_{t-i}$$

for a (learned) set of coefficients $\{c_i\}_{i=1}^\tau$ such that $a_t \approx h_\pi(s_t)$. Given feedback target $\widehat{\mathbf{A}} = \{\hat{a}_t\}$, the joint loss function thus becomes

$$\ell(a, \hat{a}_t) = \ell_d(a, \hat{a}_t) + \lambda \ell_R(a, s_t)$$
$$= (a - \hat{a}_t)^2 + \lambda (a - \sum_{i=1}^{\tau} c_i a_{t-i})^2$$

Here λ trades off between smoothness versus absolute imitation accuracy. The autoregressor h_π acts as a smooth linear regularizer, the parameters of which can be updated at each iteration based on feedback target $\widehat{\mathbf{A}}$ according to

$$h_\pi = \operatorname*{argmin}_{h \in \mathcal{H}} \left\| \widehat{\mathbf{A}} - h(\widehat{\mathbf{A}}) \right\|^2$$
$$= \operatorname*{argmin}_{c_1,\ldots,c_\tau} \left(\sum_{t=1}^{T} (\hat{a}_t - \sum_{i=1}^{\tau} c_i \hat{a}_{t-i})^2\right), \quad (17)$$

In practice we use a regularized version of equation (17) to learn a new set of coefficients $\{c_i\}_{i=1}^\tau$. The Forest procedure (Line 8 of algorithm 2) would use this updated h_π to train a new policy that optimizes the trade-off between $a_t \approx \hat{a}_t$ (feedback) versus smoothness as dictated by $a_t \approx \sum_{i=1}^{\tau} c_i a_{t-i}$.

B.1.1. Smooth Regularization with Linear Autoregressors

Our application of Algorithm 1 to realtime camera planning proceeds as follows: At each iteration, we form a state sequence $\mathbf{S}$ based on the rolled-out trajectory $\mathbf{A}$ and tracking input data $\mathbf{X}$ such that $s_t = [x_t, \ldots, x_{t-\tau}, a_{t-1}, \ldots, a_{t-\tau}]$ for appropriate τ that captures the history of the sequential decisions. We generate feedback targets $\widehat{\mathbf{A}}$ based on each $s_t \in \mathbf{S}$ following $\hat{a}_t = \sigma a_t + (1-\sigma)a_t^*$ using a parameter $\sigma \in (0,1)$ depending on the Euclidean distance between $\mathbf{A}$ and $\mathbf{A}^*$. Typically, σ gradually decreases to 0 as the rolled-out trajectory improves on the training set. After generating the targets, a new linear autoregressor h_π (new set of coefficients $\{c_i\}_{i=1}^{\tau}$) is learned based on $\widehat{\mathbf{A}}$ using regularized least squares (as described in the previous section). We then train a new model $\hat{\pi}$ based on $\mathbf{S}, \widehat{\mathbf{A}}$, and the updated coefficients $\{c_i\}$, using `Forest` - our recurrent decision tree framework that is capable of generating smooth predictions using autoregressor h_π as a smooth regularizer (see the following section for how to train smooth decision trees). Note that typically this creates a "chicken-and-egg" problem. As the newly learned policy $\hat{\pi}$ is greedily trained with respect to $\widehat{\mathbf{A}}$, the rolled-out trajectory of $\hat{\pi}$ may have a state distribution that is different from what the previously learned h_π would predict. Our approach offers two remedies to this circular problem. First, by allowing feedback signals to vary smoothly relative to the current rolled-out trajectory $\mathbf{A}$, the new policy $\hat{\pi}$ should induce a new autoregresor that is similar to previously learned h_π. Second, by interpolating distributions (Line 10 of Algorithm 2) and having $\widehat{\mathbf{A}}$ eventually converge to the original human trajectory $\mathbf{A}^*$, we will have a stable and converging state distribution, leading to a stable and converging h_π.

Throughout iterations, the linear autoregressor h_π and regularization parameter λ enforces smoothness of the rolled-out trajectory, while the recurrent decision tree framework `Forest` learns increasingly accurate imitation policy. We generally achieve a satisfactory policy after 5-10 iterations in our sport broadcasting data sets. In the following section, we describe the mechanics of our recurrent decision tree training.

B.2. Smooth Regression Tree Training

Given states s as input, a decision tree specifies a partitioning of the input state space. Let $D = \{(s_m, \hat{a}_m)\}_{m=1}^M$ denote a training set of state/target pairs. Conventional regression tree learning aims to learn a partitioning such that each leaf node, `node`, makes a constant prediction via minimizing the squared loss function:

$$\begin{aligned}\bar{a}_{\texttt{node}} &= \underset{a}{\operatorname{argmin}} \sum_{(s,\hat{a})\in D_{\texttt{node}}} \ell_d(a,\hat{a}) \\ &= \underset{a}{\operatorname{argmin}} \sum_{(s,\hat{a})\in D_{\texttt{node}}} (\hat{a}-a)^2, \qquad (18)\end{aligned}$$

where $D_{\texttt{node}}$ denotes the training data from D that has partitioned into the leaf `node`. For squared loss, we have:

$$\bar{a}_{\texttt{node}} = \texttt{mean}\left\{\hat{a}\,|(s,\hat{a}) \in D_{\texttt{node}}\right\}. \qquad (19)$$

In the recurrent extension to `Forest`, we allow the decision tree to branch on the input state s, which includes the previous predictions $a_{-1}, \ldots, a_{-\tau}$. To enforce more explicit smoothness requirements, let $h_\pi(a_{-1}, \ldots, a_{-\tau})$ denote an autoregressor that captures the temporal dynamics of π over the distribution of input sequences $d_{\mathbf{x}}$, while *ignoring* the inputs x. At time step t, h_π predicts the behavior $a_t = \pi(s_t)$ given only $a_{t-1}, \ldots, a_{t-\tau}$.

Our policy class Π of recurrent decision trees π makes smoothed predictions by regularizing the predictions to be close to its autoregressor h_π. The new loss function incorporates both the squared distance loss ℓ_d, as well as a smooth regularization loss such that:

$$\begin{aligned}\mathcal{L}_D(a) &= \sum_{(s,\hat{a})\in D} \ell_d(a,\hat{a}) + \lambda \ell_R(a,s) \\ &= \sum_{(s,\hat{a})\in D} (a-\hat{a})^2 + \lambda(y - h_\pi(s))^2\end{aligned}$$

where λ is a hyper-parameter that controls how much we care about smoothness versus absolute distance loss.

Making prediction: For any any tree/policy π, each leaf node is associated with the terminal leaf node value $\bar{a}_{\texttt{node}}$ such that prediction $\tilde{a}$ given input state s is:

$$\tilde{a}(s) \equiv \pi(s) = \underset{a}{\operatorname{argmin}}\ (a - \bar{a}_{\texttt{node}(s)})^2 + \lambda(a - h_\pi(s))^2, \qquad (20)$$

$$= \frac{\bar{a}_{\texttt{node}(s)} + \lambda h_\pi(s)}{1+\lambda}. \qquad (21)$$

where $\texttt{node}(s)$ denotes the leaf node of the decision tree that s branches to.

Setting terminal node value: Given a fixed h_π and decision tree structure, navigating through consecutive binary queries eventually yields a terminal leaf node with associated training data $D_{\texttt{node}} \subset D$.

One option is to set the terminal node value $\bar{a}_{\texttt{node}}$ to satisfy:

$$\begin{aligned}\bar{a}_{\texttt{node}} &= \underset{a}{\operatorname{argmin}} \sum_{(s,\hat{a})\in D_{\texttt{node}}} \ell_d(\tilde{a}(s|a),\hat{a}) \\ &= \underset{a}{\operatorname{argmin}} \sum_{(s,\hat{a})\in D_{\texttt{node}}} (\tilde{a}(s|a)-\hat{a})^2 \qquad (22) \\ &= \underset{a}{\operatorname{argmin}} \sum_{(s,\hat{a})\in D_{\texttt{node}}} \left(\frac{a+\lambda h_\pi(s)}{1+\lambda} - \hat{a}\right)^2\end{aligned}$$

for $\tilde{a}(s|a)$ defined as in (21) with $a \equiv \bar{a}_{\texttt{node}(s)}$. Similar to (19), we can write the closed-form solution of (22) as:

$$\bar{a}_{\texttt{node}} = \texttt{mean}\left\{(1+\lambda)\hat{a} - \lambda h_\pi(s)\,|(s,\hat{a}) \in D_{\texttt{node}}\right\}. \quad (23)$$

When $\lambda = 0$, (23) reduces to (19).

Note that (22) only looks at imitation loss ℓ_d, but not smoothness loss ℓ_R. Alternatively in the case of joint imitation and smoothness loss, the terminal leaf node is set to minimize the joint loss function:

$$\begin{aligned}
\bar{a}_{\texttt{node}} &= \operatorname*{argmin}_{a} \mathcal{L}_{D_{\texttt{node}}}(\tilde{a}(s|a)) \\
&= \operatorname*{argmin}_{a} \sum_{(s,\hat{a})\in D_{\texttt{node}}} \ell_d(\tilde{a}(s|a),\hat{a}) + \lambda\ell_R(\tilde{a}(s|a),s) \\
&= \operatorname*{argmin}_{a} \sum_{(s,\hat{a})\in D_{\texttt{node}}} (\tilde{a}(s|a)-\hat{a})^2 + \lambda(\tilde{a}(s|a)-h_\pi(s))^2
\end{aligned} \tag{24}$$

$$\begin{aligned}
&= \operatorname*{argmin}_{a} \sum_{(s,\hat{a})\in D_{\texttt{node}}} \left(\frac{a+\lambda h_\pi(s)}{1+\lambda} - \hat{a}\right)^2 \\
&+ \lambda\left(\frac{a+\lambda h_\pi(s)}{1+\lambda} - h_\pi(s)\right)^2 \\
&= \texttt{mean}\left\{\hat{a}\,|(s,\hat{a})\in D_{\texttt{node}}\right\},
\end{aligned} \tag{25}$$

Node splitting mechanism: For a node representing a subset $D_{\texttt{node}}$ of the training data, the node impurity is defined as:

$$\begin{aligned}
I_{\texttt{node}} &= \mathcal{L}_{D_{\texttt{node}}}(\bar{a}_{\texttt{node}}) \\
&= \sum_{(s,\hat{a})\in D_{\texttt{node}}} \ell_d(\bar{a}_{\texttt{node}},\hat{a}) + \lambda\ell_R(\bar{a}_{\texttt{node}},s) \\
&= \sum_{(s,\hat{a})\in D_{\texttt{node}}} (\bar{a}_{\texttt{node}}-\hat{a})^2 + \lambda(\bar{a}_{\texttt{node}}-h_\pi(s))^2
\end{aligned}$$

where $\bar{a}_{\texttt{node}}$ is set according to equation (23) or (25) over $(s,\hat{a})$'s in $D_{\texttt{node}}$. At each possible splitting point where $D_{\texttt{node}}$ is partitioned into $D_{\texttt{left}}$ and $D_{\texttt{right}}$, the impurity of the left and right child of the node is defined similarly. As with normal decision trees, the best splitting point is chosen as one that maximizes the impurity reduction: $I_{\texttt{node}} - \frac{|D_{\texttt{left}}|}{|D_{\texttt{node}}|}I_{\texttt{left}} - \frac{|D_{\texttt{right}}|}{|D_{\texttt{node}}|}I_{\texttt{right}}$