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:*

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

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

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

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:

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

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

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

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

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 $\mathbb{E}_{\tilde{\boldsymbol{A}}}[\tilde{a}_t] = a_t\ \forall t = 1, \ldots, T$, where the expectation is taken over all random roll-outs of π_{sto}.*

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

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 $\epsilon = \max_{s\sim d_\pi} \|\hat{\pi}(s) - \pi(s)\|$. 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:

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

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 $x_1, \ldots, x_T$.

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

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

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

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:

Claim: $\|s'_t - s_t\| \leqslant \frac{\beta\epsilon}{(1-\beta)(1-\gamma)}$

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 $\left\|s'_{t-1} - s_{t-1}\right\| \leqslant$

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

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

□

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

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

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 $\ell(\pi'(s'_t)) - \ell(\pi(s_t)) = \ell(\pi'(s'_t)) - \ell(\pi'(s_t)) + \ell(\pi'(s_t)) - \ell(\pi(s_t)) \leqslant L\|\pi'(s'_t) - \pi'(s_t)\| + \beta(\ell(\hat{\pi}(s_t)) - \ell(\pi(s_t)))$ due to convexity and L-Lipschitz continuity of ℓ.

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

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

(by condition 1 and definition of ϵ).

From here, one can use this recursive relation to easily show that $\|s'_t - s_t\| \leqslant \beta\epsilon t$ 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. $\omega = \min_{f\in\mathcal{F}} \mathbb{E}_{x\sim\mathcal{X}}[\|f([x,0]) - a^*\|]$. 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.:*

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

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, $f_0 = \operatorname*{argmin}_{f\in\mathcal{F}} \sum_{t=1}^T \|f([x_t,0]) - a_t^*\|^2$. 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 $\mathcal{H}$.

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

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

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

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

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 $a_t \approx \sum_{i=1}^{\tau} c_i a_{t-i}$.

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 τ, 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: $\mathbf{A}_n = \{a_t^n\} \leftarrow \{\pi_{n-1}([x_{t:t-\tau}, a^{n-1}_{t-1:t-\tau}])\}$ *//sequential roll-out old policy*
5: $\mathbf{S}_n \leftarrow \{s_t^n = [x_{t:t-\tau}, a^n_{t-1:t-\tau}]\}$ *//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: $h_n = \operatorname*{argmin}_{c_1, \ldots, c_\tau} \sum_{t=1}^{T} \left(\hat{a}_t^n - \sum_{i=1}^{\tau} c_i \hat{a}^n_{t-i}\right)^2$ *//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 β to weighted empirical errors*
10: $\pi_n = \beta \hat{\pi}_n + (1-\beta)\pi_{n-1}$ *// update policy*
11: **end for**

output Last policy π_N

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 se-

quence $\mathbf{S}$ 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:

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

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

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

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:

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

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

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:

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

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:

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

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: $I_{\texttt{node}} - \frac{|D_{\texttt{left}}|}{|D_{\texttt{node}}|} I_{\texttt{left}} - \frac{|D_{\texttt{right}}|}{|D_{\texttt{node}}|} I_{\texttt{right}}$