Online No-regret Model-Based Meta RL for Personalized Navigation

Yuda Song [1] YUDAS@ANDREW.CMU.EDU
Ye Yuan [1] YYUAN2@CS.CMU.EDU
Wen Sun [2] WS455@CORNELL.EDU
Kris Kitani [1] KKITANI@CS.CMU.EDU
[1] *Carnegie Mellon University,* [2] *Cornell University*

Editors: R. Firoozi, N. Mehr, E. Yel, R. Antonova, J. Bohg, M. Schwager, M. Kochenderfer

Abstract

The interaction between a vehicle navigation system and the driver of the vehicle can be formulated as a model-based reinforcement learning problem, where the navigation systems (agent) must quickly adapt to the characteristics of the driver (environmental dynamics) to provide the best sequence of turn-by-turn driving instructions. Most modern day navigation systems (e.g, Google maps, Waze, Garmin) are not designed to personalize their low-level interactions for individual users across a wide range of driving styles (e.g., vehicle type, reaction time, level of expertise). Towards the development of personalized navigation systems that adapt to a variety of driving styles, we propose an online no-regret model-based RL method that quickly conforms to the dynamics of the current user. As the user interacts with it, the navigation system quickly builds a user-specific model, from which navigation commands are optimized using model predictive control. By personalizing the policy in this way, our method is able to give well-timed driving instructions that match the user's dynamics. Our theoretical analysis shows that our method is a no-regret algorithm and we provide the convergence rate in the agnostic setting. Our empirical analysis with 60+ hours of real-world user data using a driving simulator shows that our method can reduce the number of collisions by more than 60%. The full version of the paper can be found at https://arxiv.org/abs/2204.01925.

Keywords: Model-based RL, Online Learning, System Identification, Personalization

1. Introduction

Reinforcement Learning (RL) and Markov Decision Processes (MDP) Sutton and Barto (2018) provide a natural paradigm for training an agent to build a navigation system. That is, by treating the navigation system that delivers instructions as the *agent* and combining the vehicle and the user as the *environment*, we can design RL algorithms to train a navigation agent that provides proper instructions by maximizing a reward of interest. For example, we could reward the agent when it delivers accurate and timely instructions. There has been a rich line of research on applying RL in various navigation tasks, including robot navigation Surmann et al. (2020); Liu et al. (2020), indoor navigation Chen et al. (2020a, 2021a), navigation for blind people OhnBar et al. (2018), etc. In this work, we put our focus on vehicle navigation, a task also explored by the previous literature Kiran et al. (2021); Stafylopatis and Blekas (1998); Deshpande and Spalanzani (2019).

However, most of the previous works only consider a static scenario, where the user (robot, vehicle or human) that receives the instructions does not change its dynamics during the deployment phases. This is not practical in the real world - in practice, the navigation system is faced with

Figure 1: Different participants using different driving kits in our navigation system built on Carla. Each participant at least interacts with the system for three hours.

Figure 2: Examples of dynamics changes: left: using different vehicles (cybertruck) and different visual situations (mimicking foggy situations). Right: using a first-person view instead of a third-person view, while the latter is easier to control.

different users with changing or unseen dynamics during deployment (in our case, we could face different drivers, vehicles and weather conditions everyday), which requires fast personalization of the navigation system. However, there is little hope learning a monotonic policy that fits all (and unseen) users. In fact, even well-established navigation systems such as Google Maps are only designed for the average user. Thus, personalization needs to be achieved through adaptation, and we need a policy that can quickly adapt to the incoming user. Recently, meta learning Finn et al. (2017a); Duan et al. (2016); Finn et al. (2017b) presents a way to train policies that adapt to new tasks by few-shot adaptation. Similarly, in our online setting, we need to focus on sample efficiency in order to have performance guarantee, i.e., we want to adapt with as few samples as possible - otherwise, a policy is sub-optimal as long as the adaptation is unfinished. Towards this direction, model-based RL methods have shown promising sample complexity/efficiency results (compared to model-free algorithms) in both theoretical analysis Sun et al. (2019); Tu and Recht (2019) and empirical applications Chua et al. (2018); Wang et al. (2019).

Drawing the above intuitions from meta RL and MBRL, in this work we propose an online model-based meta RL algorithm for building a personalized vehicle navigation system. Our algorithm trains a meta dynamics model that can quickly adapt to the ground truth dynamics of each incoming user, and a policy can easily be induced by planning with the adapted model. Theoretically, we show that our algorithm is no-regret. Empirically, we develop a navigation system inside the Carla simulator Dosovitskiy et al. (2017) to evaluate our algorithm. The state space of the navigation system is the specification of the vehicle (e.g., location, velocity) and the action space is a set of audio instructions. A policy will be trained and tested inside the system with a variety of scenarios (driver, vehicle type, visual condition, level of skills). Ideally, a personalized policy should deliver timely audio instructions to each different user. We conduct extensive experiments (60+ hours) on real human participants with diverse driving experiences as shown in Fig. 1, which is significantly more challenging than most previous works that only perform experiments with simulated agents instead of real humans Koh et al. (2020); Nagabandi et al. (2018b). The user study shows that our algorithm outperforms the baselines in following optimal routes and it is better at enforcing safety where the collision rate is significantly reduced by more than 60%. We further highlight that the proposed algorithm is able to generate new actions to facilitate the navigation process.

Our contributions are as follows: 1) we propose an algorithm that guarantees an adaptive policy that achieves near-optimal performance in a stream of changing dynamics and environments. 2) Theoretically, we prove our algorithm is no-regret. 3) We build a vehicle navigation system inside the state-of-the-art driving simulator that could also facilitate future research in this direction. 4) We

perform extensive real-human experiments which demonstrate our method's strong ability to adapt to user dynamics and improve safety.

2. Related Works

2.1. RL for Navigation System

The reinforcement learning paradigm fits naturally into the navigation tasks and has encouraged a wide range of practical studies. In this work we focus on vehicle navigation Koh et al. (2020); Kiran et al. (2021); Stafylopatis and Blekas (1998); Deshpande and Spalanzani (2019). Previous work can be separated into two major categories: navigation with human interaction Thomaz et al. (2006); Wang et al. (2003); Hemminahaus and Kopp (2017) and navigation without human interaction Kahn et al. (2018); Ross et al. (2008). The first category includes tasks such as indoor navigation and blind person navigation. Such tasks require considering personalization issues when facing different users, who potentially bring drastically different dynamics to the system. Previously, vehicle navigation is categorized into the second field since there is no human interaction. However, in our work, we provide a novel perspective and introduce new challenges by incorporating human drivers into the system. Thus in order for an algorithm to solve this new task, it should consider personalization, accuracy and safety at the same time. In this work, we theoretically and empirically show that our proposed solution could address the three key components simultaneously.

2.2. Model-based RL and Meta RL

Model-based RL and meta RL are designed to achieve efficient adaptation. MBRL aims to model the dynamics of the environments and plan in the learned dynamics models. Prior works Sun et al. (2019); Tu and Recht (2019); Chua et al. (2018); Song and Sun (2021) have shown superior sample efficiency of model-based approaches than their model-free counterparts, both in theory and in practice. For navigation with human interaction, prior works have shown promising results by modeling human dynamics Torrey et al. (2013); Iqbal et al. (2016); Daniele et al. (2017). However, these works do not provide any personalized navigation.

Towards the adaptability side, meta RL approaches Finn et al. (2017a,b); Duan et al. (2016) have demonstrated the ability to adapt to a new task during test time by few-shot adaptation. Previously, there are also works that propose meta-learning algorithms for model-based RL Nagabandi et al. (2018b,a); Lin et al. (2020); Clavera et al. (2018); Belkhale et al. (2021); Sæmundsson et al. (2018). However, these methods do not have any theoretical guarantee. Additionally, it is unclear if these works can adapt in real-world scenarios since their evaluation is based only on simulation or non-human experiments. In contrast, the proposed method is evaluated with extensive user study (60+ hours) with real-human participants which demonstrates the method's ability to deliver personalized driving instructions that quickly adapts to the ever-changing dynamics of the user.

3. Preliminaries

A finite horizon (undiscounted) Markov Decision Processes (MDP) is defined by $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P, H, C, \mu\}$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transitional dynamics kernel, $C : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the cost function, H is the horizon and $\mu \in \Delta(\mathcal{S})$ is the initial state distribution. In this work, we consider an online learning setup where we encounter a stream

of MDPs $\{\mathcal{M}^{(t)}\}_{t=1}^{T}$. Each $\mathcal{M}^{(t)}$ is defined by $\{\mathcal{S}, \mathcal{A}, P^{(t)}, H, C, \mu\}$, i.e., the transitional dynamics kernels are different across the MDPs. The goal is to learn a stochastic policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that minimizes the total sum of costs. Denote $d^{\pi}_{h,\mu,P^{(t)}}(s,a) = \mathbb{E}_{s\sim\mu}\left[\mathbb{P}(s_h = s, a_h = a | s_0 = x, \pi, P^{(t)})\right]$ to be the probability of visiting (s,a) under policy π, initial distribution μ and transition $P^{(t)}$. Let $d^{\pi}_{\mu,P^{(t)}} = \frac{1}{H}\sum_{h=0}^{H-1} d^{\pi}_{\mu,h,P^{(t)}}$ be the average state-action distribution. Since in the online setting, if the algorithm incurs a sequence of (adapted) policies $\boldsymbol{\pi} = \{\pi^{(t)}\}_{t=1}^{T}$, the value at round t is $V^{(t)}(s) = \mathbb{E}\left[\sum_{h=0}^{H-1} C(s_h, a_h | s_0 = s, \pi^{(t)}, P^{(t)})\right]$. Then $J(\boldsymbol{\pi}) = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{s\sim\mu}\left[V^{(t)}(s)\right]$ is the objective function. We also consider the best policy from hindsight to be adaptive: let $\boldsymbol{\pi}^{*} = \{\pi^{*(t)}\}_{t=1}^{T}$ be a sequence of policies adapted from a meta policy (or model in MBRL setting) π^{*} (which we will define rigorously in Sec. 5), and we mildly abuse notation here by letting Π be the class of regular policies and meta policies, then we define the regret of an algorithm up to time T as: $R(T) = J(\boldsymbol{\pi}) - \min_{\pi^{*}\in\Pi} J(\boldsymbol{\pi}^{*})$, and the goal is to devise an algorithm whose regret diminishes with respect to T.

Finally we introduce the coverage coefficient term for a policy π: $c^{\pi}_{\nu} = \sup_{s,a}\frac{d^{\pi}_{\mu}(s,a)}{\nu(s,a)}$, where ν is some state-action distribution. This is the maximum mismatch between an exploration distribution ν and the state-action distribution induced by a policy π.

4. Online Meta Model-based RL

In this section we present our algorithm: Online Meta Model-based Reinforcement Learning, a combination of online model-based system identification paradigm and meta model-based reinforcement learning. We provide the pseudocode of our algorithm in Alg.1, Appendix Sec.B.

Our overall goal is to learn a meta dynamics model $\widehat{P} : \mathcal{S}\times\mathcal{A} \to \Delta(\mathcal{S})$ in an online manner, such that if we have access to a few samples $D = \{s_d, a_d, s'_d\}_{d=1}^{|D|} \sim d^{\pi}_{P^{(t)}}$ induced by some policy π in the environment of interest $\mathcal{M}^{(t)}$, we can quickly adapt to the new environment by performing one-shot gradient descent on the model-learning loss function ℓ (such as MLE): $U(\widehat{P}, D) := \widehat{P} - \alpha_{\text{adapt}}\frac{1}{|D|}\sum_{d=1}^{|D|}\nabla\ell(\widehat{P}(s_d, a_d), s'_d)$, which will be an accurate estimation on the ground truth dynamics $P^{(t)}$. Here the gradient is taken with respect to the parameters of $\widehat{P}$ and α_{adapt} is the adaptation learning rate. This remains for the rest of this section. With such models, we then can plan (running policy optimization methods or using optimal control methods) to obtain a policy for each incoming environment.

Specifically, we assume that we begin with an exploration policy π^{e} which could be a predefined non-learning-based policy. Such an exploration policy is important in two ways: first is to collect samples for the adaptation process and second is to cover the visiting distribution for a good policy for the theoretical analysis. We also assume we have access to an offline data set D_{off} which includes transition samples from a set of MDPs $\{\mathcal{M}_j\}_{j=1}^{M}$, where M should be much smaller than T. In practice, we can simply build the offline dataset by rolling out the explore policy in $\{\mathcal{M}_j\}_{j=1}^{M}$. Our algorithm starts with first training a warm-started model $\widehat{P}^{(1)}$ with the offline dataset:

$$\widehat{P}^{(1)} = \operatorname*{argmax}_{P\in\mathcal{P}} \sum_{s,a,s'\in D_{\text{off}}} \log P(s'|s,a), \tag{1}$$

where $\mathcal{P}$ is the model class.

Next during the online phase when every iteration t a new $\mathcal{M}^{(t)}$ comes in, at the beginning of this iteration we first collect one trajectory τ_t with the exploration policy π^e, then we perform one shot adaptation on our latest meta model $\widehat{P}^{(t)}$ to obtain $U(\widehat{P}^{(t)}), \tau_t)$:

$$U(\widehat{P}^{(t)}, \tau_t) = \widehat{P}^{(t)} + \alpha_{\text{adapt}} \nabla \sum_{s,a,s' \in \tau_t} \log(\widehat{P}^{(t)}(s'|s,a)). \tag{2}$$

After we have the estimation of the current dynamics, we can construct our policy $\widehat{\pi}^{(t)}$, which is defined by running model predictive control methods such as CEM De Boer et al. (2005), MPPI Williams et al. (2017) or tree search Schrittwieser et al. (2020) on the current adapted dynamics model $U(\widehat{P}^{(t)}, \tau_t)$. Thus we denote the policy as $\widehat{\pi}^{(t)} = \text{MPC}(U(\widehat{P}^{(t)}, \tau_t))$.

With the constructed policy $\widehat{\pi}^{(t)}$, we then collect K trajectories inside the current $\mathcal{M}^{(t)}$: with probability $\frac{1}{2}$ we will follow the current policy $\widehat{\pi}^{(t)}$, and with probability $\frac{1}{2}$ we will follow the exploration policy π^e.[1] For simplicity, we denote the state-action distribution induced by the exploration policy as $\nu_t := d^{\pi^e}_{P^{(t)}}$, which will be used for analysis in the next section. Finally after we store all the trajectories we obtained in the current iteration into dataset D_t, along with samples from previous iterations, we can perform no-regret online meta learning algorithm (e.g., Follow The Meta Leader) to obtain the new meta model $\widehat{P}^{(t+1)}$:

$$\widehat{P}^{(t+1)} = \operatorname*{argmax}_{P \in \mathcal{P}} \sum_{n=1}^{t} \sum_{s,a,s' \in D_n} \log U(P, \tau_n)(s'|s,a). \tag{3}$$

Here we treat the model class and the meta model class as the same class since they are defined to have the same parametrization.

Thus the proposed method integrates online learning, meta learning and MBRL and is better than the previous algorithms in the following sense: comparing with online SystemID methods, our method can return a set of personalized policies when incoming environments have different dynamics rather than policy for one fixed environment. Comparing with meta MBRL methods, our method can learn to adapt in an *online* manner under extensive real-human user-study, where previous methods only work in simulation. Also meta RL needs to be trained on a set of tasks before testing so it is not applicable for our online setting. Another advantage of our method is that it has theoretical guarantee comparing with the meta RL methods, which we will show in the next section.

5. Analysis

In this section we provide the regret analysis for Alg. 1. At a high level, our regret analysis is based on the previous analysis of the online SystemID framework Ross and Bagnell (2012) that applies DAgger Ross et al. (2011) in the MBRL setting to train the sequence of models, where DAgger can be regarded as a no-regret online learning procedure such as Follow-the-Leader (FTL) algorithm Shalev-Shwartz et al. (2011). In our setting, on the other hand, our model training framework could be regarded as Follow-the-Meta-Leader (FTML) Finn et al. (2019) that is also no-regret with our designed training objective. The overall intuition is that with the sequence of more reliable adapted models, the regret can be bounded and diminish over time. We first introduce our assumptions:

1. Note that the probability $\frac{1}{2}$ here is for the ease of theoretical analysis. In fact, one can change the probability of rolling out with π^e to arbitrary non-zero probability and will only change a constant coefficient on the final result.

Assumption 1 (Optimal Control Oracle) *We assume that the model predictive control methods return us an $\epsilon -$ optimal policy. That is, for any transitional dynamics kernel P and initial state distribution μ, we have*

$$\mathbb{E}_{s\sim\mu}\left[V^{P,MPC(P)}(s)\right] - \min_{\pi\in\Pi}\mathbb{E}_{s\sim\mu}\left[V^{P,\pi}(s)\right] \leq \epsilon_{oc}.$$

Assumption 2 (Agnostic) *We assume agnostic setting. That is, our model class may not contain the ground truth meta model that can perfectly adapt to any dynamics. Formally, given arbitrary state-action distribution d, one trajectory τ induced by $P^{(t)}$, $\exists\widehat{P}^* \in \mathcal{P}$, such that $\forall t \in [T]$,*

$$\mathbb{E}_{s,a\sim d}\left[D_{KL}(U(\widehat{P}^*,\tau)(\cdot|s,a), P^{(t)}(\cdot|s,a))\right] \leq \epsilon_{model}.$$

Note that ϵ_{oc} and ϵ_{model} are independent from our algorithm (i.e., we can not eliminate this term no matter how carefully we design our algorithm).

Our analysis begins with bounding the performance difference between the sequence of policies generated by our algorithm and any policy π', which can be the policy induced by adapting from the best meta-model. As the first step, we can relate the performance difference to the model error with: 1. simulation lemma (Lemma 5) Kearns and Singh (2002); Sun et al. (2019). 2. We relate the trajectories induced by π' and π^e by the coefficient term introduced in the Sec. 3. Note that since we should only care about comparing with "good" π', the coefficient is small under such circumstances.

The next step is thus to bound the model error over the online process. Let's first observe the following loss function: $\ell^{(t)}(P) = \mathbb{E}_{s,a\sim\rho_t}\left[D_{KL}\left(P(\cdot|s,a), P^{(t)}(\cdot|s,a)\right)\right]$, where ρ_t defined as the state-action distribution of our data collection scheme. This correspond to the model error. Then if we use FTML to update the meta model: $\widehat{P}^{(t+1)} = \text{argmin}_{P\in\mathcal{P}}\sum_{i=1}^{t}\ell^{(t)}(U(P,\tau_t))$, it is easy to see this update is equivalent to Eq. 3 as in our algorithm. Then we can invoke the main result from Finn et al. (2019) (Lemma 8), and we are ready to present our main result:

Theorem 3 *Let $\{\widehat{P}^{(t)}\}_{t=1}^T$ be the learned meta models. Let $\{U(\widehat{P}^{(t)},\tau_t)\}_{t=1}^T$ be the adapted model after the one-shot adaptations. Let $\widehat{\boldsymbol{\pi}} = \{\widehat{\pi}^{(t)}\}_{t=1}^T$, where $\widehat{\pi}^{(t)} := MPC(U(\widehat{P}^{(t)},\tau_t))$. Let $\rho_t := \frac{1}{2}d_{P^{(t)}}^{\widehat{\pi}^{(t)}} + \frac{1}{2}\nu_t$ be the state-action distribution induced by our algorithm under $\mathcal{M}^{(t)}$. Then for policy sequence $\boldsymbol{\pi'} = \{\pi'^{(t)}\}_{t=1}^T$, we have* [2]

$$J_\mu(\widehat{\boldsymbol{\pi}}) - J_\mu(\boldsymbol{\pi'}) \leq \epsilon_{oc} + \max_t(c_{\nu_t}^{\pi'^{(t)}})H^2\sqrt{\epsilon_{model}} + \tilde{O}\left(\frac{\max(c_{\nu_t}^{\pi'^{(t)}})H^2}{\sqrt{T}}\right).$$

We defer the full proofs in Appendix A. Thm. 3 shows that given any $\boldsymbol{\pi'}$, which could be the policies induced by the best meta model in the model class, the performance gap incurred by our algorithm diminishes with rate $\tilde{O}(\frac{1}{\sqrt{T}})$, along with the inevitable terms ϵ_{oc} and ϵ_{model}, which our algorithm has no control over. Since our main result is based on any arbitrary policy sequence, we here first formally define the best adapted sequence of policies from hindsight, and we could easily conclude with the no-regret conclusion in the following corollary:

2. Note that $\tilde{O}$ hides the logarithmic terms.

Corollary 4 *Let $\widehat{\boldsymbol{\pi}} = \{\widehat{\pi}^{(t)}\}_{t=1}^{T}$ be the policies returned by our algorithm. Define $\pi_P^* = \operatorname{argmin}_{\pi\in\Pi} \mathbb{E}_{s\sim\mu} V^{\pi,P}(s)$. Let $\boldsymbol{\pi}_P^* = \{\pi^*_{U(P,\tau_t)}\}_{t=1}^{T}$. For simplicity let $\pi^{*(t)} = \pi^*_{U(P,\tau_t)}$. Let $R(T) = J(\widehat{\boldsymbol{\pi}}) - \min_{P\in\mathcal{P}} J(\boldsymbol{\pi}_P^*)$ be the regret of our algorithm. We have*

$$\lim_{T\to\infty} R(T) = \epsilon_{oc} + \max_t (c_{\nu_t}^{\pi^{*(t)}}) H^2 \sqrt{\epsilon_{model}}.$$

6. Experiments

6.1. System Setup

Our vehicle navigation system is developed inside the Carla simulator Dosovitskiy et al. (2017). Each navigation run follows the below procedure: a vehicle spawns at a specified location inside the simulator, and the navigation system provides one audio instruction to the user every one second (45 frames). The user will control the vehicle with either a keyboard or a driving kit, and react to the audio instructions, until the vehicle is near the specified destination point. The goal is to compare the reliability of different navigation algorithms based on optimal path tracking and safety.

We develop the navigation system in 5 Carla native town maps, which include comprehensive traffic situations. We use Carla's built-in A* algorithm to search for the optimal path: we first manually specify a starting point and destination point, which define a route, and the A* planner returns a list of waypoints, defined by its x,y,z coordinates, that compose the optimal path.

To introduce various and comprehensive dynamics, we invite human participants with different levels of skills and propose three ways to induce diverse dynamics: a) different built-in vehicles, b) different visual conditions, c) different cameras (points of view), as shown in Fig. 2.

Finally, we introduce our design for the state and action spaces for our RL setting. We discretize the system by setting every 45 frames (around 1s) as one time step. We design each route to have a length of 2 minutes on average, thus each trajectory contains 120 state-action pairs on average. Each state is a 14-dimensional vector that contains low-level information about the vehicle, user inputs, and the immediate goal. For the action space, we adopt a discrete action space given the limited audio generation and modification functionality of Carla's interface. Nevertheless, we observe that our current design suffices to deliver timely instructions combined with our proposed algorithm. In fact, our algorithm could also expand the action space, which we will demonstrate in detail in Sec 6.4. Concretely, one action corresponds to one audio instruction, and the action space contains the 7 instructions in total, including the no instruction option. In practice, we only play the corresponding audio when the current action is different from the previous action, or the current action has been repeated for 15 timesteps.

We defer a complete list of system details in Appendix C: the town maps can be found in Sec. C.1 , state representation in Sec. C.2 , actions in Sec. C.3, list of different vision conditions, vehicles, driving kits in Sec. C.4.

6.2. Practical Implementation

In this section we introduce the practical implementation details. There are two major specifications for a successful transfer from theory to practice: the model parametrization and the MPC method.

For the model parametrization, we represent the (meta) model class $\mathcal{P}$ as a class of deep feedforward neural networks. Each element in the class can be denoted as P_θ, where here θ is the parameters of P. Here we adopt a deterministic model given there is no stochasticity in

the underlying Carla system. Furthermore, since some instructions have lasting affect on system dynamics (e.g., prepare to turn), we feed a history of state-action pairs to the network for robust predictions. Concretely, the network input is a history of state-action pairs $h_t = \{s_{t-k}, a_{t-k}\}_{k=0}^{K}$ and the output of the network is the predicted next state $\widehat{s}_{t+1}$. In practice using $K = 4$ suffices to include enough information for model prediction. In addition, we discover that predicting state differences in the egocentric vehicle coordinates boosts the accuracy of the model.

Because of the deterministic system, maximizing log-likelihood is equivalent to minimizing the l_2 norms of the state differences. In this work, we follow the empirical success of L-step loss Nagabandi et al. (2018b); Luo et al. (2018). Specifically, with a batch $\{h_t, s_{t+1}\}_{t=1}^{B}$, where B is the batch size, for P_θ, we update the model by

$$\theta := \theta - \alpha_{\text{meta}} \sum_{i=1}^{B} \nabla_\theta \left(\sum_{l=1}^{L} \left\| (\hat{s}_{i+l} - \hat{s}_{i+l-1}) - (s_{i+l} - s_{i+l-1}) \right\|_2 \right), \tag{4}$$

where $\hat{h}_i = h_i$ and $\hat{s}_{i+l+1} = U(P_\theta(\hat{h}_{i+l}, a_{i+l}))$, and $\hat{h}_{i+l+1}$ is constructed by discarded the first state action pair in $\hat{h}_{i+l+1}$ and concatenate $\hat{s}_{i+l+1}$ and a_{i+l+1} to its end. In our experiments, we use $L = 5$. Similarly, the parameter of $U(P_\theta)$ is obtained by using Eq. 4 with the adaptation sample (line 3 in Alg. 1) where next states are predicted by P_θ.

For MPC, since the action space is discrete, we adopt a tree search scheme, although the current design can be easily switched to MPPI or CEM. We define the cost of an action sequence $\{a_h\}_{h=1}^{H}$ under P_θ by: $C(\{a_h\}_{h=1}^{H}, P_\theta) = \sum_{h=1}^{H} \|\hat{s}_{h+1} - g_{h+1}\|_2^2$, where $\{\hat{s}_{h+1}\}_{h=1}^{H}$ are obtained by autoregressively rolling out P_θ with $\{a_h\}_{h=1}^{H}$ and $\{g_{h+1}\}_{h=1}^{H}$ are the coordinates of goal waypoints. Note on the right hand side we abuse notation by letting $\hat{s}$ also be coordinates (but not the full states). Intuitively, we want the model-induced policy to be a residual policy of the explore policy, since the explore policy is accurate (but not personalized enough) for most situations. Thus we reward an action sequence that is the same as the explore policy action sequence (but at potentially shifted time intervals). We also add penalties for turning actions where there are no junctions.

6.3. Experimental Results

We evaluate our practical algorithm inside the system described in Sec. 6.1. For each of the 5 towns, we design 10 routes that contain all traffic conditions. We define using one navigation algorithm to run the 10 routes in one town as one iteration. For each of the towns, we first use the explore policy to collect 3 iterations of samples as the offline dataset. For the online process, the first 7 iterations involve different vehicles and visual conditions. In 3 out of 5 towns, we invite additional 6 participants with diverse driving skills to complete the next 6 iterations. Then for all the maps, we include 2 additional iterations that are controlled by different control equipment. Each human participant contributes at least 3-hour driving inside the system, and in total more than 60-hour driving data is used for our final evaluation.

We include three baselines in this study: the first is the *explore policy*[3], a non-learning-based policy based on A* path planning algorithm. The underlying backbone of the explore policy is the Carla's autopilot agent, which provides expert demonstrations for the state-of-the-art autonomous

3. The name "explore policy" inherits from Ross and Bagnell (2012), but here the explore policy is actually a very strong baseline, as described in the main text.

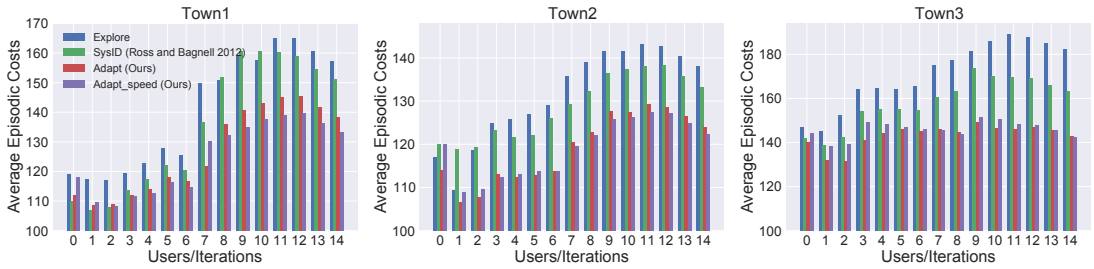

Figure 3: Performance bar plots of the four baselines in three town maps. The metric we use is the average episodic accumulative costs upon the current iteration. Here the episodic accumulative costs are calculated by averaging the point-wise tracking error over the ten routes that we defined in Sec. 6.1

driving algorithms Chen et al. (2020b, 2021b); Prakash et al. (2021). To increase the accuracy and robustness of the policy, we further include hard-coded heuristics for the corner cases. The second baseline is a policy induced by the non-adaptive model trained online Ross and Bagnell (2012). We refer to this policy as *SysID* in the rest of the sections. The model training of SysID follows Eq. 4 with only one exception that the next states are predicted directly by the models instead of the adapted models (i.e., no meta adaptation). The third baseline is our algorithm and we include a variation of our algorithm as well, which will be introduced in detail in Sec. 6.4.

The metric we use in this section is the point-wise tracking error. Concretely, for a state s_t, the cost we are evaluating is $c(s_t) = \|x_{s_t} - x_{g_{t-1}}, y_{s_t} - y_{g_{t-1}}\|_2^2$, where x, y denotes the coordinates elements inside a state and g_{t-1} denotes the goal selected from the waypoint list from the last timestep. We follow Sec. 3 and plot the empirical performance $\hat{J}(\boldsymbol{\pi}) = \frac{1}{T}\sum_{t=1}^{T} H_t^{\pi^{(t)}}$, where H_t is the sum of costs in one episode. We show the performance of the 4 baselines in 3 town maps that involve human participants with different levels of driving skills in Fig. 3 (we defer the results of the remaining two maps in Appendix D). We observe that except for the first online iteration, our method (denoted as adapt) beats the other two baselines (explore and SysID) consistently in tracking the optimal routes. There are two possible reasons that the navigation system could accumulate huge costs: an ill-timed instruction could cause the drive to miss a turn at a junction thus follow a suboptimal route. On the other hand, if the navigation system is not adaptive to the driver, mildly ill-timed instructions may still keep the driver on the optimal route, but deviate from the waypoints (e.g., driving on the sidewalk). This will also raise safety issues as we will further evaluate in Sec. 6.5. Note that our results indicate the significance of personalization: non-adaptive methods, even with online updates (SysID), behave suboptimally and we observe the performance gaps between non-adaptive methods and our method increase as the more users are involved in the system. We also observe that dynamics change induced by different users is much more challenging than simulation-based dynamics change as real human participants introduce significantly greater dynamics shifts than simulation-based changes, as indicated in Fig.3 starting from user/iteration 7.

6.4. Generating New Actions

In this section we analyze a new variant of our adaptive policy: adaptive policy with speed change. The new policy is designed to generate new actions by changing the playback speed of the original instructions: slower, normal and faster. Thus the new policy increases the action space dimensions by three times. We include the performance of this new variant along with other baselines in

Fig. 3, denoted as "Adaptive_speed". We see that this new policy design performs similarly to the original adaptive policy at the first few iterations due to the lack of samples with speed change, but it gradually outperforms as the online procedure goes on.

6.5. Safety

In addition to better tracking, our empirical evaluation also shows that our method has better safety guarantees. In this section, we present the number of collisions for each method in Table 1. In each cell we show the total number of collisions averaged over the 10 routes in the corresponding town maps. We then average over the 15 online iterations and present the mean and standard deviation. We observe that comparing with the explore policy baseline, our methods (adaptive and adaptive speed) achieve over 50% and 60% decrease in terms of the number of collisions, respectively. The results indicate that by better tracking the optimal path and delivering more timely instructions with personalized policies, the adaptive policies indeed achieve a stronger safety guarantee.

6.6. Empirical Regret

In this section, we analyze the empirical regret of the adaptive policy, i.e., how our adaptive policy compares with the policy induced by the best empirical meta model from hindsight. We train the hindsight policy using the data collected from all 15 iterations and train with Eq. 3. We compare the two policies in Fig. 4, following the same metric as in Fig. 3. The results show that the performance gap exists at the beginning, but it closes up as the online process proceeds. Here we want to emphasize that in practice, this empirical regret (performance gap) may not diminish strictly at a rate of $T^{-1/2}$ since we could only expect such rate in expectation, we do observe that such gap decreases over time and in some iteration, the performance of the adaptive policy is actually the same as the best policy from hindsight, indicating the strong adaptability of our proposed method.

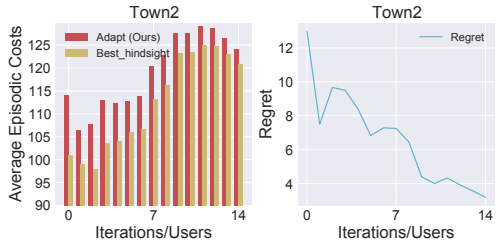

Figure 4: Evaluation of empirical regret/performance gap of the adaptive policy in town 2. The evaluation metric and presentation follow Fig. 3.

	Town1	Town2	Town3
Explore policy	3.08 (1.73)	1.00 (1.18)	1.38 (1.21)
SysID	2.23 (1.19)	0.69 (1.14)	1.23 (1.19)
Adapt (Ours)	1.85 (1.29)	0.31 (0.61)	0.69 (0.72)
Adapt Speed (Ours)	1.54 (1.45)	0.23 (0.42)	0.54 (0.50)

Table 1: Average counts of total collisions of each baseline in the 3 town maps. The average is over 15 online iterations and the standard deviation is indicated in the parenthesis.

7. Summary

In this work we propose an online meta MBRL algorithm that can adapt to different incoming dynamics in an online fashion. We proved that our algorithm is no-regret. We specifically study the application of the proposed algorithm in personalized voice navigation and develop a machine learning system, which could also be beneficial for future research. The extensive real-world use study shows that our algorithm could indeed efficiently adapt to unseen dynamics, and also improve the safety of the voice navigation.

References

Suneel Belkhale, Rachel Li, Gregory Kahn, Rowan McAllister, Roberto Calandra, and Sergey Levine. Model-based meta-reinforcement learning for flight with suspended payloads. *IEEE Robotics and Automation Letters*, 6(2):1471–1478, 2021.

Changan Chen, Unnat Jain, Carl Schissler, Sebastia Vicenc Amengual Gari, Ziad Al-Halah, Vamsi Krishna Ithapu, Philip Robinson, and Kristen Grauman. Soundspaces: Audio-visual navigation in 3d environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 17–36. Springer, 2020a.

Changan Chen, Sagnik Majumder, Ziad Al-Halah, Ruohan Gao, Santhosh Kumar Ramakrishnan, and Kristen Grauman. Learning to set waypoints for audio-visual navigation. *arXiv preprint arXiv:2008.09622*, 2020b.

Changan Chen, Ziad Al-Halah, and Kristen Grauman. Semantic audio-visual navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15516–15525, 2021a.

Dian Chen, Vladlen Koltun, and Philipp Krähenbühl. Learning to drive from a world on rails. *arXiv preprint arXiv:2105.00636*, 2021b.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. In *Conference on Robot Learning*, pages 617–629. PMLR, 2018.

Andrea F Daniele, Mohit Bansal, and Matthew R Walter. Navigational instruction generation as inverse reinforcement learning with neural machine translation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 109–118. IEEE, 2017.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

Niranjan Deshpande and Anne Spalanzani. Deep reinforcement learning based vehicle navigation amongst pedestrians using a grid-based state representation. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2081–2086. IEEE, 2019.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl ^{2}: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017a.

Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pages 357–368. PMLR, 2017b.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning*, pages 1920–1930. PMLR, 2019.

Jacaueline Hemminahaus and Stefan Kopp. Towards adaptive social behavior generation for assistive robots using reinforcement learning. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 332–340. IEEE, 2017.

Tariq Iqbal, Maryam Moosaei, and Laurel D Riek. Tempo adaptation and anticipation methods for human-robot teams. In *RSS, Planning HRI: Shared Autonomy Collab. Robot. Workshop*, 2016.

Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5129–5136. IEEE, 2018.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2):209–232, 2002.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

Songsang Koh, Bo Zhou, Hui Fang, Po Yang, Zaili Yang, Qiang Yang, Lin Guan, and Zhigang Ji. Real-time deep reinforcement learning based vehicle navigation. *Applied Soft Computing*, 96: 106694, 2020.

Zichuan Lin, Garrett Thomas, Guangwen Yang, and Tengyu Ma. Model-based adversarial meta-reinforcement learning. *arXiv preprint arXiv:2006.08875*, 2020.

Lucia Liu, Daniel Dugas, Gianluca Cesari, Roland Siegwart, and Renaud Dubé. Robot navigation in crowded environments using deep reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5671–5677. IEEE, 2020.

Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. *arXiv preprint arXiv:1807.03858*, 2018.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018a.

Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based rl. *arXiv preprint arXiv:1812.07671*, 2018b.

Eshed OhnBar, Kris Kitani, and Chieko Asakawa. Personalized dynamics models for adaptive assistive navigation systems. In *Conference on Robot Learning*, pages 16–39. PMLR, 2018.

Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7077–7087, 2021.

Stephane Ross and J Andrew Bagnell. Agnostic system identification for model-based reinforcement learning. *arXiv preprint arXiv:1203.1007*, 2012.

Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayesian reinforcement learning in continuous pomdps with application to robot navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 2845–2851. IEEE, 2008.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. Meta reinforcement learning with latent variable gaussian processes. *arXiv preprint arXiv:1803.07551*, 2018.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011.

Yuda Song and Wen Sun. Pc-mlp: Model-based reinforcement learning with policy cover guided exploration. In *International Conference on Machine Learning*, pages 9801–9811. PMLR, 2021.

Andreas Stafylopatis and Konstantinos Blekas. Autonomous vehicle navigation using evolutionary reinforcement learning. *European Journal of Operational Research*, 108(2):306–318, 1998.

Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches. In *Conference on learning theory*, pages 2898–2933. PMLR, 2019.

Hartmut Surmann, Christian Jestel, Robin Marchel, Franziska Musberg, Houssem Elhadj, and Mahbube Ardani. Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments. *arXiv preprint arXiv:2005.13857*, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Andrea Lockerd Thomaz, Cynthia Breazeal, et al. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *Aaai*, volume 6, pages 1000–1005. Boston, MA, 2006.

Cristen Torrey, Susan R Fussell, and Sara Kiesler. How a robot should give advice. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 275–282. IEEE, 2013.

Stephen Tu and Benjamin Recht. The gap between model-based and model-free methods on the linear quadratic regulator: An asymptotic viewpoint. In *Conference on Learning Theory*, pages 3036–3083. PMLR, 2019.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *CoRR*, abs/1907.02057, 2019. URL http://arxiv.org/abs/1907.02057.

Yi Wang, Manfred Huber, Vinay N Papudesi, and Diane J Cook. User-guided reinforcement learning of robot assistive tasks for an intelligent environment. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 1, pages 424–429. IEEE, 2003.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.