CS329D Final Project: Covariate shifts in multi-agent interactions

Robin Brown
rabrown1@stanford.edu

Robert Dyro
rdyro@stanford.edu

Rohan Sinha
rhnsinha@stanford.edu

Abstract

In recent years, applications in autonomous driving have motivated a significant research effort on sequential decision making agents that need to interact competitively, yet safely, with other agents. In this project, we consider a simple police pursuit setting; the objective is for the ego vehicle to close in as tightly as possible on the opponent without being rammed. While the ego vehicle has observed a history of safe interactions, closing the gap with the opponent necessitates venturing into regions of the state space that have not yet been observed. In essence, inducing a covariate shift between previously observed trajectories, and future trajectories. In contrast to many of the examples discussed in class, the covariate shift is not prescribed by the environment, but rather controlled by the ego agent. In order for the ego agent to take more aggressive, yet safe, actions requires tightly bounding the probability of collision of a function of the covariate shift. In this project, we consider two scenarios: in the first, the ego agent fits a model to the opponent's known policy class. We then apply the exact uncertainty model towards trajectory optimization using a particle approach. In the second, the ego agent fits a model that is agnostic to the opponent's policy class, removing our ability to characterize epistemic uncertainty, so we constrain the policy updates to be small. We find that the exact model class and particle MPC approach was able to achieve strong performance. In contrast, the agnostic model class typically resulted in infeasible optimization problems. Our results demonstrate the importance of using *all* knowledge of problem structure when incorporating knowledge of covariate shifts.

1 Introduction

To reliably deploy the next generation of autonomous robotic systems in unstructured open-world environments, they must safely interact with other agents such as humans and other robots. In many of these settings, such as when an autonomous vehicle must merge into traffic on a highway on-ramp, the autonomous agent cannot communicate with other agents to form a collaborative strategy. More importantly, automated systems will often have to compete with other agents to force their objective. In the autonomous driving industry, interaction with other agents is typically managed using trajectory forecasting methods that predict the trajectories of nearby agents. Given these predictions, the planning and control stack then constructs a motion plan for the AV to avoid collisions. However, such a naive approach does not take the influence of the ego's motion plan on the other agents into account.

Therefore, this requires the development of strategies that intelligently reason about the effect that the decisions made by an automated robot have on the actions of the agents in its environment. Typically the behavior of other agents is unknown and needs to be learned from data. However, an autonomous agent cannot freely explore to learn about their opponents, as collisions and other dangerous events should be strictly avoided. In this work, we propose to study interaction aware methods with safety

Preprint. Under review.

guarantees in robotics by focusing on an autonomous police pursuit problem between two self-driving cars.

We chose to pursue this problem because it represents an interesting scenario where the ego agent is in control of the covariate shift: an update to the ego agent's policy shifts the distribution of joint trajectories of all the agents in a scene. This contrasts with many of the examples presented in class, where the environment dictates the covariate shift. In our setting, the ego agent must produce a model of the opposing agent's control policy, and act within it's safety constraints (i.e., the probability of collision cannot exceed a certain threshold). We expect that producing competitive behavior necessitates problem specific modelling to tightly bound the probability of collision as a function of covariate shift. Similarly, we expect that when the model is agnostic to the opponent's policy class, the resulting bounds become too conservative for the ego agent to deviate from previously observed trajectories.

2 Related Work

At a high-level, we segment existing methods for interaction-aware planning into two categories:

Ego-conditioned Trajectory Forecasting: One solution is to predict the joint evolution of the trajectories of all the agents in a scene [1] or explicitly condition forecasts of other agents' trajectories on the motion plans of the ego vehicle [2], thereby capturing the joint interaction between the agents. By conditioning the predictions on the decisions made by the ego vehicle in the planning stack, as done in [3], these methods can reason about how others will react to the decisions made by the autonomous vehicle (AV).

Modeling the opponent's policy: Instead of forecasting the trajectories of others, other work models the opponent's policy directly, for example by directly fitting a parametric model of the opponent's policy to data [4]. This contrasts with game-theoretic approaches, such as [5, 6, 7], that anticipate an opponent's actions by assuming knowledge of their objective. Combining game-theoretic iterative best-response algorithms with inverse reinforcement learning methods [8] to infer the objectives of others was investigated in [9]. However, solving for a Nash equilibrium is generally computationally intractable and these approaches do not take uncertainty on the learned objective into account. It is unclear whether a certainty equivalent Nash-equilibrium found is still valid when noise is factored in.

If we have an explicit model of an opponent's reactive decision making, we can reduce the multi-agent control problem to a single-agent planning problem. This is because with an explicit opponent model, the outcome of the multi-agent interaction is completely determined by the ego's decision making. Intuitively, updating the ego policy causes a covariate shift on the trajectory distribution at deployment, therefore, we need to account for the deterioration in model performance to avoid dangerous situations. Single agent planning under explicit representations of uncertainty in the dynamics model parameters has become the core premise of the learning-based control (i.e., see [10, 11, 12]) and model-based reinforcement learning (i.e., see [13, 14]) communities, resulting in methods that implicitly consider the covariate shift induced by policy updates because they have access to a globally accurate uncertainty estimates in the dynamics. In this work, we first consider applying the particle MPC algorithm [12] to the reduced single-agent planning problem. This algorithm takes a scenario approach [15] to optimize the trajectory of an uncertain system with safety guarantees. However, oftentimes, like when we train deep trajectory forecasting models or apply IRL, we do not have access to a distribution on the true parameters of the prediction model. Therefore, we also develop analysis to explicitly extract the intuition that we need to constrain the policy updates to be "small" to retain confidence in a model's predictions.

Treating uncertainty in sequential or repeated decision-making problems through the lens of covariate shift has been investigated in the literature previously to some extent [16, 17, 18]. For example, in a single agent RL setting, Schulman et. al. compute trust-regions to constrain policy updates to improve robustness [16] and in [18], the authors derive bounds on the modelling error in a repeated static decision making problem, though neither approaches arrive at safety guarantees. At present, we are not aware of methods that exploit the multi-agent nature of the problem we consider in a model-based setting.

3 Set-up

In this paper, we consider a simple police pursuit example, where the ego ("police") vehicle is following behind the opponent's vehicle. The opponent's strategy is to follow a nominal velocity, but

it attempts to ram the ego vehicle if it gets too close. The objective is for the ego vehicle to close in as tightly as possible on the opponent without getting rammed.

Formally, our example is modelled by an environment with three states, $x = [\Delta p, v_{\text{ego}}, v_{\text{op}}]^T$, where Δp is the distance between the ego and opponent, and v_{ego}, v_{op} are the velocities of the ego and opponent, respectively. The controls, $u = [u_{\text{ego}}, u_{\text{op}}]$, represent the accelerations of both agents, and are bounded by

$$\left|u_{\text{ego}}^{(i)}\right| \leq l_{\text{ego}} \quad |u_{\text{op}}| \leq l_{\text{op}}$$

where $l_{\text{ego}} < l_{\text{op}}$. The opponent's policy is modelled by a feedback law,

$$u_{\text{op}}^{(i)} = \pi_{\text{op}}\big(x^{(i)}; \theta\big) = -\theta_1(v_{\text{op}}^{(i)} - v_{\text{ref}}) - \theta_2 \frac{1}{\Delta p}.$$

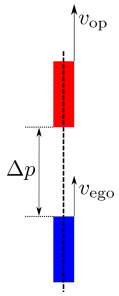


Figure 1: Toy example

We assume process noise in the controls is modelled by a zero-mean normal distribution with variance σ^2. Thus, the dynamics of the system are

$$x^{(i+1)} = \begin{bmatrix} \Delta p^{(i+1)} \\ v_{\text{ego}}^{(i+1)} \\ v_{\text{op}}^{(i+1)} \end{bmatrix} = \begin{bmatrix} \Delta p^{(i)} - T(v_{\text{ego}}^{(i)} - v_{\text{op}}^{(i)}) \\ v_{\text{ego}}^{(i+1)} + T(u_{\text{ego}}^{(i)} + \epsilon_{\text{ego}}^{(i)}) \\ v_{\text{op}}^{(i)} + T(u_{\text{op}}^{(i)} + \epsilon_{\text{op}}^{(i)}) \end{bmatrix} =: f(x^{(i)}, u_{\text{ego}}^{(i)}, \pi_{\text{op}}(x^{(i)}), \tag{1}$$

with $\epsilon_*^{(i)} \sim \mathcal{N}(0, \sigma^2)$. A collision is defined by the event $\Delta p^{(i)} < 0$. Our objective is to find an input trajectory for $u_{\text{ego}} = \pi\big(x^{(i)}\big)$ that minimizes Δp, without inducing a collision. Specifically, we want to design a controller which solves the following optimal control problem over a horizon of T timesteps

$$\begin{aligned} \pi_{\text{ego}}^*(x) = \underset{\pi_{\text{ego}}(\cdot)}{\arg\min} \quad & \mathbb{E}\big[\sum_{i=1}^{T} \Delta p^{(i)}\big] \\ \text{subject to} \quad & x^{(i+1)} = \begin{bmatrix} \Delta p^{(i)} - T(v_{\text{ego}}^{(i)} - v_{\text{op}}^{(i)}) \\ v_{\text{ego}}^{(i+1)} + T(\pi_{\text{ego}}\big(x^{(i)}; \theta\big) + \epsilon_{\text{ego}}^{(i)}) \\ v_{\text{op}}^{(i)} + T(\pi_{\text{op}}\big(x^{(i)}; \theta\big) + \epsilon_{\text{op}}^{(i)}) \end{bmatrix}, \\ & \mathbb{P}[\Delta p^{(i)} \geq 0, \forall i] \geq 1 - \delta, \\ & \left|u_{\text{ego}}^{(i)}\right| \leq l_{\text{ego}}, |u_{\text{op}}| \leq l_{\text{op}}. \end{aligned} \tag{2}$$

Problem (2) optimizes the system trajectory, subject to a chance constraint that specifies our risk tolerance for a collision. The dynamics (1) and the trajectory optimization problem (2) show that our control problem reduces to a single-agent planning problem if we knew the policy of the opponent. Instead, we approximate the policy of the opponent from data, and need to take our modeling uncertainty into account.

4 Opponent Models

We consider two methods of modelling the opponent. Firstly, we apply ordinary least-squares to identify the opponents' policy. Then, we consider arbitrary approaches to model the uncertain closed loop dynamics (1).

4.1 Modelling the Opponent's Policy—Exact Hypothesis Class

In this section, we consider the setting where the model class of the opponent's policy is known. In particular, we seek to fit the three parameters, θ_1, θ_2 and V_{ref}, given observations of $x^{(i)}$.

At each time step, an estimate of the opponent's control action, $\hat{u}_{\text{op}}^i$, is computed as

$$\hat{u}_{\text{op}}^{(i)} = \frac{1}{T}(v_{\text{op}}^{(i+1)} - v_{\text{op}}^{(i)}) \tag{3}$$

Given a trajectory up to time step K, the ordinary least squares regression of the parameters are then given by

$$\hat{\beta} = (X^\top X)^{-1} X^\top Y, \quad X = \begin{bmatrix} v_{op}^{(0)} & \frac{1}{\Delta p^{(0)}} & 1 \\ v_{op}^{(1)} & \frac{1}{\Delta p^{(1)}} & 1 \\ \vdots & \vdots & \vdots \\ v_{op}^{(K)} & \frac{1}{\Delta p^{(K)}} & 1 \end{bmatrix}, \quad Y = \begin{bmatrix} \hat{u}_{op}^{(0)} \\ \hat{u}_{op}^{(1)} \\ \vdots \\ \hat{u}_{op}^{(K)} \end{bmatrix} \tag{4}$$

and $\theta_1 = -\beta_1,\ \theta_2 = -\beta_2$ and $v_{\text{ref}} = \frac{\beta_3}{\beta_1}$.

Given a future time step, the predicted controls are given by $\hat{u} = \hat{\beta}^\top x^{(i)}$, while the true controls as given by $u = \beta^\top x^{(i)} + \epsilon$. By the standard bias-variance decomposition of the OLS estimator, the expected error in the control estimates is given by,

$$\begin{aligned} \mathbb{E}_Y[\hat{u} - u \mid X, x^{(i)}] &= \mathbb{E}_Y[\hat{\beta}^\top x^{(i)} - (\beta^\top x^{(i)} + \epsilon) \mid X, x^{(i)}] \\ &= \mathbb{E}_Y[((X^\top X)^{-1} X^\top (X\beta + \epsilon_X))^\top x^{(i)} - (\beta^\top x^{(i)} + \epsilon) \mid X, x^{(i)}] \\ &= \mathbb{E}_Y[(\beta^\top x^{(i)} - \beta^\top x^{(i)}) + ((X^\top X)^{-1} X^\top \epsilon_X)^\top x^{(i)} + \epsilon] \\ &= 0, \end{aligned}$$

and it's variance is

$$\begin{aligned} \mathbb{E}_Y[(\hat{u} - u)^2 \mid X, x^{(i)}] &= \mathbb{E}[(((X^\top X)^{-1} X^\top \epsilon_X)^\top x^{(i)} - \epsilon)^2 \mid X, x^{(i)}] \\ &= \mathbb{E}[(((X^\top X)^{-1} X^\top \epsilon_X)^\top x^{(i)})^2 \mid X, x^{(i)}] \\ &\quad - 2\mathbb{E}[\epsilon((X^\top X)^{-1} X^\top \epsilon_X)^\top x^{(i)}) \mid X, x^{(i)}] \\ &\quad + \mathbb{E}[\epsilon^2] \\ &= \sigma^2(1 + x^{(i),\top}(X^\top X)^{-1} x^{(i)}), \end{aligned} \tag{5}$$

showing us the predictive quality deteriorates when we stray from the training data. In particular, the OLS estimator yields a distribution over the parameter β as

$$\beta - \hat{\beta} \sim \mathcal{N}(0, \sigma^2 (X^\top X)^{-1}). \tag{6}$$

We can then use the model distribution to safely plan the ego trajectory under model uncertainty.

4.2 Modelling the Opponent's Policy—Agnostic Hypothesis Class

Besides the well-specified OLS setting, we consider a setting where the model class of the opponent's policy is unknown and we cannot quantify the uncertainty on the model parameters. To do this, we simply ignore the uncertainty on the parameters of the OLS estimate (6) so we can construct a strong baseline. Other common approaches include Gaussian Process regression [19], by fitting a prediction model on historical trajectory data and using it in planning as in [13, 10], or fitting deep neural networks [14]. While models like GPs are able to represent *aleatoric* uncertainty on the closed-loop trajectories, resulting from the process noise ϵ, using an agnostic model removes our ability to consider *epistemic* uncertainty on the parameters that results from the fact that our model is innacurate far from the training data.

5 Particle MPC

In the OLS setting, we apply the particle model predictive control (PMPC) algorithm to optimize the trajectory of the [12]. The particle MPC algorithm extends standard methods for nonlinear trajectory optimization based on sequential convex programming (SCP) to account for model uncertainty using a monte-carlo sampling approach. To optimize the chance constrained trajectory optimization problem (2), the particle MPC algorithm samples M parameters from the uncertainty model over the dynamics parameters in (6), $\{\beta_k\}_{k=1}^M \overset{\text{iid}}{\sim} \mathcal{N}(\hat{\beta}, \sigma^2(X^\top X)^{-1})$ and propagates the dynamics $x_k^{(i+1)} = f(x^{(t)}, u_{\text{ego}}^{(i)}; \beta_k)$ forwards in the optimization procedure to approximate the uncertainty over trajectories induced by the model uncertainty. Similarly, particles are sampled and propagated for the process noise as well. By enforcing the collision avoidance constraint as a hard constraint on all the particles, we can approximate the true chance constraint in (2), as well as approximate

Algorithm 1: SCP PMPC

Require: Initial states $\{x_i^{(0)}\}_{i=1}^M$, dynamics models $\{f_i\}_{i=1}^M$, solution guess $\{x_i, u_i\}_{i=1}^M$
Require: Hyperparameters $\rho_x, \rho_u, N_\text{c}$
Require: Solution tolerance ϵ
repeat
$\{\bar{f}_i^{(j)}, \nabla_x f_i, \nabla_u f_i\}_{i=1}^M \leftarrow$ Linearize dynamics around the trajectory guess $\{x_i, u_i\}_{i=1}^M$;
Split the cost into the convex and non-convex parts $\{c_{i,\text{cvx}}^{(j)}, c_{i,\text{ncvx}}^{(j)}\}_{i=1}^M$;
$\{\nabla_x c_{i,\text{ncvx}}^{(j)}, \nabla_u c_{i,\text{ncvx}}^{(j)}\}_{i=1}^M \leftarrow$ Linearize non-convex cost around $\{x_i, u_i\}_{i=1}^M$;
$\{\Delta x_i^{(j)}, \Delta u_i^{(j)}\}_{i=1}^M \leftarrow$ Solution to SCP step ;
$\{x_i, u_i\}_{i=1}^M \leftarrow \{x_i + \Delta x_i, u_i + \Delta u_i\}_{i=1}^M$
until $\sum_{i=1}^M \sum_{j=0}^N \|\Delta x_i^{(j)}\| + \|\Delta u_i^{(j)}\| < \epsilon$;
return $\{x_i, u_i\}_{i=1}^M$ =0

the expected cost by averaging the cost incurred by each particle. This approach is summarized in algorithm 1, we refer readers to [12] for further algorithmic details. In addition, the particle MPC algorithm is in essence a scenario optimization algorithm, and we refer readers to [15] for results on selecting the number of particles to ensure that the chance constraint in problem (2) is satisfied.

6 Constrained Policy Updates

The goal of our work is to develop methods that reason intelligently about the covariate shifts induced by a policy update on the quality of predictive models of the behavior of other agents. This is an important problem of practical significance, because using innacurate predictions on other agents' decisions a robot's trajectory can result in dangerous situations. Intuitively, one should generally expect their behavior model's predictions to be accurate only close to the training data, and therefore constrain policy updates to be "small" enough for the predictions to remain accurate throughout the policy iteration process. In a safety-critical robotics setting, it is therefore important to carefully consider the trade-off between changing our agent's policy to improve it's performance (exploration) and retaining confidence in the outcome of the interaction (safety).

However, by specifying uncertainty models as in sections 4.1, and applying nonconvex trajectory solvers that account for model uncertainty like the particle MPC algorithm as presented in section 5, this trade-off is implicitly (and optimally) managed by the trajectory optimization solver: For the OLS estimator we discussed in section 4.1, equation (5) shows that the uncertainty in the predictions is a function of the state at which the prediction is made. By propagating the statistical uncertainty on the learned model parameters, the particle MPC already takes this additional uncertainty into account. In the most general setting, for example when we use deep generative neural networks for trajectory forecasting as in [1] or use neural networks to represent the objectives of other agents [9], we do not have access to posterior distributions over the model parameters, or the model may be misspecified. Therefore, uncertainty aware planning algorithms cannot account for the deterioration in the prediction quality that results from a covariate shift induced by a policy update in this setting. Therefore, we present a generic analysis of the multi-agent trajectory optimization task in this section, and use it to inform practical control design.

Consider a trajectory optimization problem on a horizon of length N. The ego robot has state $x \in \mathbb{R}^n$ and takes inputs $u \in \mathbb{R}^m$. The opponent has state $y \in \mathbb{R}^n$ and takes inputs $z \in \mathbb{R}^m$. For simplicity, assume that both agents follow the same (possibly stochastic) dynamics

$$x_{t+1} \sim f(x_t, u_t), \quad y_{t+1} \sim f(y_t, z_t). \tag{7}$$

We start by assuming that the opponent acts according to some policy on the joint state, i.e., that the opponents actions satisfy some $\pi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m$. This yields interactive behavior between the agents: the opponent conditions its behavior on both its position and the ego's. We want to guarantee that the agents avoid a collision with high probability. To represent this, let

$$\mathbf{x} = [x_0^\top, \dots, x_N^\top]^\top$$
$$\mathbf{y} = [y_0^\top, \dots, y_N^\top]^\top$$

be the trajectories of the ego and the opponent. Then, let $\mathcal{A} \subseteq \mathbb{R}^{Nn} \times \mathbb{R}^{Nn}$ be the set of trajectories where the ego and the opponent collide. For example, in our police pursuit example, the collisions are determined by the relative position of the agents as $\mathcal{A} = \{\mathbf{x}, \mathbf{y} : \exists t \in \{0, \dots, N\} \text{ s.t. } p_t^1 - p_t^2 > 0\}$. For another example, when both agents occupy euclidean balls of radius $\epsilon/2$ around their respective state, we get collision avoidance constraints (the complement of $\mathcal{A}$) of the form

$$\|x_t - y_t\| \geq \epsilon, \quad \forall t = 0, \dots, N.$$

When we design the ego policy, we need to satisfy the collision avoidance chance constraint with risk tolerance δ,

$$\mathbb{P}_{p(\mathbf{x},\mathbf{y})}\big((\mathbf{x}, \mathbf{y}) \in \mathcal{A}\big) \leq \delta. \tag{8}$$

For shorthand notation, let $A(\mathbf{x}, \mathbf{y})$ be the event that the collision avoidance constraint is **not** satisfied over the trajectory horizon, that is, when $(\mathbf{x}, \mathbf{y}) \in \mathcal{A}$. Through our trajectory optimization procedure, we get to decide $p(\mathbf{x})$. Since the opponent makes decisions based on $\mathbf{x}$, the marginal distribution $p(\mathbf{y})$ is not independent of $\mathbf{x}$. But, since we know $p(\mathbf{x})$, we can factor the joint distribution as

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x}). \tag{9}$$

However, the behavior of the opponent, $p(\mathbf{y}|\mathbf{x})$, is unknown. Instead, we approximate it using a model with parameters θ: $p_\theta(\mathbf{y}|\mathbf{x}) \approx p(\mathbf{y}|\mathbf{x})$. So, if we design $p(\mathbf{x})$, we have the approximate joint distribution $p_\theta(\mathbf{x}, \mathbf{y}) := p_\theta(\mathbf{y}|\mathbf{x})p(\mathbf{x})$. As before, we consider a *policy iteration* setting, so that θ was learned from a trajectory dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ collected using a different trajectory distribution $p'(\mathbf{x})$.

Note that we can use the particle MPC algorithm to enforce the chance constraint under the approximate joint distribution p_θ. To conservatively enforce the chance constraint on the *true joint distribution* using the *approximate joint distribution*, we introduce the following lemma.

Lemma 1. *Let $p_\theta(\mathbf{y}|\mathbf{x})$ be the trajectory forecasting model parameterized by θ and fix the ego trajectory distribution $p(x)$. Then, if*

$$\mathbb{P}_{p_\theta(\mathbf{x},\mathbf{y})}(A(\mathbf{x}, \mathbf{y})) + \text{bound}(\theta, p(\mathbf{x}), p(\mathbf{y}|\mathbf{x})) \leq \delta, \tag{10}$$

it holds that $\mathbb{P}_{p(\mathbf{x},\mathbf{y})}(A(\mathbf{x}, \mathbf{y})) \leq \delta$. Here, we define

$$\text{bound}(\theta, p(\mathbf{x}), p(\mathbf{y}|\mathbf{x})) := \sqrt{\frac{1}{2}\mathbb{E}_{p(\mathbf{x})}\big[\mathbb{D}_{\text{KL}}(p(\mathbf{y}|\mathbf{x})||p_\theta(\mathbf{y}|\mathbf{x}))\big]}. \tag{11}$$

Proof. To construct the bound, notice that

$$\begin{aligned}
\big|\mathbb{P}_{p(\mathbf{x},\mathbf{y})}(A(\mathbf{x}, \mathbf{y})) - \mathbb{P}_{p_\theta(\mathbf{x},\mathbf{y})}(A(\mathbf{x}, \mathbf{y}))\big| &= \big|\mathbb{E}_{p(\mathbf{x})}[\mathbb{P}_{p(\mathbf{y}|\mathbf{x})}(A)] - \mathbb{E}_{p(\mathbf{x})}[\mathbb{P}_{p_\theta(\mathbf{y}|\mathbf{x})}(A)]\big| \\
&\leq \mathbb{E}_{p(\mathbf{x})}\big[|\mathbb{P}_{p(\mathbf{y}|\mathbf{x})}(A) - \mathbb{P}_{p_\theta(\mathbf{y}|\mathbf{x})}(A)|\big] && \text{(Jensen's inequality)} \\
&\leq \mathbb{E}_{p(\mathbf{x})}\big[\text{TV}\big(p(\mathbf{y}|\mathbf{x}), p_\theta(\mathbf{y}|\mathbf{x})\big)\big] && \text{(Def. of Total Variation Distance)} \\
&\leq \mathbb{E}_{p(\mathbf{x})}\big[\sqrt{\frac{1}{2}\mathbb{D}_{\text{KL}}(p(\mathbf{y}|\mathbf{x})||p_\theta(\mathbf{y}|\mathbf{x}))}\big] && \text{(Pinsker's inequality)} \\
&\leq \sqrt{\frac{1}{2}\mathbb{E}_{p(\mathbf{x})}\big[\mathbb{D}_{\text{KL}}(p(\mathbf{y}|\mathbf{x})||p_\theta(\mathbf{y}|\mathbf{x}))\big]} && \text{(Concave Jensen's).}
\end{aligned}$$

Using the bound, we see that

$$\begin{aligned}
\mathbb{P}_{p(\mathbf{x},\mathbf{y})}(A(\mathbf{x}, \mathbf{y})) &\leq \mathbb{P}_{p_\theta(\mathbf{x},\mathbf{y})}(A(\mathbf{x}, \mathbf{y})) + |\mathbb{P}_{p(\mathbf{x},\mathbf{y})}(A(\mathbf{x}, \mathbf{y})) - \mathbb{P}_{p_\theta(\mathbf{x},\mathbf{y})}(A(\mathbf{x}, \mathbf{y}))| \\
&\leq \mathbb{P}_{p_\theta(\mathbf{x},\mathbf{y})}(A(\mathbf{x}, \mathbf{y})) + \text{bound}(\theta, p(\mathbf{x}), p(\mathbf{y}|\mathbf{x})).
\end{aligned}$$

Therefore, enforcing the conservative constraint (10) ensures the collision avoidance constraint (8) is satisfied. □

The lemma states that if we know an upper bound on the expectation in equation (11), then we can guarantee the robots do not crash at runtime with probability at least $1 - \delta$. However, we cannot evaluate the bound, as we do not know $p(\mathbf{y}|\mathbf{x})$, that's the whole point of this analysis. Instead, we learn p_θ with trajectory data that we collected when we deployed the robot using a different ego policy $p'(\mathbf{x})$. The core issue is that $p_\theta(\mathbf{y}|\mathbf{x})$ won't approximate $p(\mathbf{y}|\mathbf{x})$ well when we stray into regions of the state space where $p'(\mathbf{x})$ did not wander. However, notice that a typical objective when we regress

a distribution is to minimize the KL divergence between the true and the learned distribution. That is, when we fit $p_\theta(\mathbf{y}|\mathbf{x})$, we approximately solve

$$\min_{\theta\in\Theta} \big\{ L_{p'}(\theta) := \mathbb{E}_{p'(\mathbf{x})}\big[\mathbb{D}_{\mathrm{KL}}(p(\mathbf{y}|\mathbf{x})||p_\theta(\mathbf{y}|\mathbf{x}))\big]\big\}, \tag{12}$$

through empirical risk minimization by minimizing the negative log-likelihood of the model over the training data. Therefore, to instantiate the bound in lemma 1, we relate expected loss of the model under the *old policy* $p'(\mathbf{x})$ that we used for data-collection to the expected loss of the *new policy* $p(\mathbf{x})$. To do this, we make use of the variational representation of the KL-divergence.

Theorem 1. *[Donsker and Varadan's variational representation [20, 21]] Let p and p' be two probability measures on the sample space $\mathcal{X}$. Then, it holds that*

$$\mathbb{D}_{\mathrm{KL}}(p||p') = \sup_{g:\mathcal{X}\to\mathbb{R}} \mathbb{E}_p[g(x)] - \log \mathbb{E}_{p'}[e^{g(x)}]. \tag{13}$$

As a direct corollary of theorem 1, we can bound the expectation in the bound (11), that is $L_p(\theta)$, using knowledge of the old policy.

Corollary 1. *When we fix the function $g(\mathbf{x},\mathbf{y})$ in theorem 1 as the loss function $\ell(\mathbf{x},\mathbf{y};\theta) = \log(\frac{p(\mathbf{y}|\mathbf{x})}{p_\theta(\mathbf{y}|\mathbf{x})})$, it follows from theorem 1 that*

$$L_p(\theta) := \mathbb{E}_{p(\mathbf{x})}\big[\mathbb{D}_{\mathrm{KL}}(p(\mathbf{y}|\mathbf{x})||p_\theta(\mathbf{y}|\mathbf{x}))\big] \le \mathbb{D}_{\mathrm{KL}}(p(\mathbf{x})||p'(\mathbf{x})) + \log \mathbb{E}_{p'(\mathbf{x},\mathbf{y})}[e^{\ell(\mathbf{x},\mathbf{y};\theta))}], \tag{14}$$

since the covariate shift setting implies that $\mathbb{D}_{\mathrm{KL}}(p(\mathbf{x})||p'(\mathbf{x})) = \mathbb{D}_{\mathrm{KL}}(p(\mathbf{x},\mathbf{y})||p'(\mathbf{x},\mathbf{y}))$.

Therefore, we can essentially bound $L_p(\theta)$ using a quantity that we can estimate from the dataset $\mathcal{D}$ sampled using p'; and the shift between the *old trajectory* and the *new trajectory*, which we can control when we run a trajectory optimization routine. Therefore, (14) and lemma 1 make it explicit that we need to limit the divergence between policy updates to maintain confidence that we will not cause a collision. Alternatively, if we merely considered the negative log-likelihood of the model and not the likelihood ratio as the loss function, the conditional entropy of $\mathbf{y}$ given $\mathbf{x}$ under p' would additively appear in the bound (14). In both cases, this constitutes a term that we would need to estimate from an empirical sample.

Instead, we note that if we are able to exactly learn $p(\mathbf{y}|\mathbf{x})$ over the support of the old policy $p'(\mathbf{x})$, i.e., when $L_{p'}(\theta) = 0$, this implies that the log term in (14) vanishes. In this case, lemma 1 shows us that we would only need to tighten the chance constraint to account for the shift in the ego-trajectories. Under such an assumption, if we constrained our policy update to lie in the KL-ball

$$\mathcal{P} = \{p(\mathbf{x}) : \mathbb{D}_{\mathrm{KL}}(p(\mathbf{x})||p'(\mathbf{x})) \le \rho\},$$

then lemma 1 gives us that enforcing the conservative chance constraint

$$\mathbb{P}_{p_\theta(\mathbf{x},\mathbf{y})}(A(\mathbf{x},\mathbf{y})) \le \delta - \sqrt{\rho/2}$$

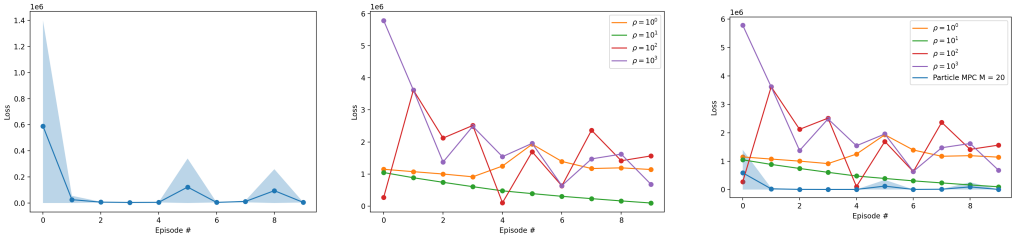
in a particle MPC problem, ensures the true chance constraint (8) will be satisfied upon deployment of the policy. In our experiments, we use this approach as a practical method to constrain our policy updates to be "small enough" so that we improve the safety of the system. We consider a complete theoretical treatment, applying uniform convergence bounds and fully instantiating the bounds we discussed as the main direction for future work. In addition, we note that we can cast the problem of upper bounding $L_p(\theta)$ through lens of distributionally robust optimization [22, 23], as

$$L_p(\theta) \le \sup_{p\in\mathcal{P}} \mathbb{E}_{p(\mathbf{x})}\big[\mathbb{D}_{\mathrm{KL}}(p(\mathbf{y}|\mathbf{x})||p_\theta(\mathbf{y}|\mathbf{x}))\big],$$

suggesting distributionally robust training of the forecasting model as an interesting direction of future work.

7 Empirical Results

In this section, we seek to evaluate the extent to which the exact and agnostic hypothesis classes allow for exploration while counterbalancing the safety requirements. We evaluate both hypothesis classes with two means of instantiating the chance constraints: (1) using particle MPC and (2) through the heuristic developed in Section 6. In particle the MPC setting, we instantiate the chance constraints by sampling from our uncertainty set, and enforcing the constraint that the appropriate percentage of



(a) Baseline Particle MPC Approach (b) ρ Constrained Strategy (c) Comparison

Figure 2: Loss evolution per episode in episodic RL. Particle MPC approach improves quickly after the first episode, whereas the performance of the ρ constrained approach varies depending on the selection of the ρ value. All ρ approaches perform worse than the Particle MPC baseline.

particles satisfy the safety constraints. In the heuristic setting, we leverage the result of Lemma 1, which states that true probability of collision is bounded by the probability of collision in the learned model and the KL divergence between the learned and true distributions. However, since the true distribution is unknown, the bound provided in Lemma 1 is not directly actionable. Instead, we leverage the fact that $p_\theta(\mathbf{y}|\mathbf{x})$ should approximate $p(\mathbf{y}|\mathbf{x})$ well in the regions of the state-space that have been observed, i.e., $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, and enforce an ℓ_2 bound on the difference previously observed trajectories and future trajectories as a proxy for constraining the divergence between trajectory distributions resulting from the process noise.

We found the nonlinear trajectory optimization to be highly numerically unstable and sensitive to hyperparameters settings. For instance, we also tried representing the agnostic hypothesis class as a GP, yielded small fit error to the predicted dynamics' history, but ultimately resulted in infeasible optimization problems, even with smoothness enforced. Figure 2 shows the major results of trajectory optimization and compares the two methods. We found that the particle MPC approach was able to improve rapidly after only one iteration. In contrast, the ρ-constrained loss consistently had worse loss than the particle MPC. We found that an intermediate value of $\rho = 10$ led to the most consistent decrease in loss in each subsequent iteration. Intuitively, this result is not surprising since the parameter ρ is meant to strike a balance between exploration and exploitation. The experiments confirm that if we make ρ too small, the system is unable to explore and improve its performance. Conversely, if ρ is too large e.g., when $\rho = 1e3$, the robot explores in regions where the model confidence is low, resulting in high trajectory cost resulting from constraint violations especially when little data is available.

8 Discussion and Conclusions

In this project, we have constructed a police pursuit toy problem to study the role of covariate shifts in principled navigation of the exploration-exploitation trade-off under safety constraints. We modelled the opposing agent both using the exact hypothesis and an agnostic hypothesis class. Given both of these models, we also solved our MPC problem using a particle MPC approach, and with a heuristic based on Lemma 1. We found that the agnostic hypothesis class often resulted in unfeasible optimization problems, despite having low loss on the fitted points. We also found that the heuristic approach resulted in far higher losses than the particle MPC approach. In conjunction, our results underscore the importance of utilizing *all* knowledge of the problem structure; as a rule of thumb, we found that generality came at the expense of being too conservative to be practically useful. Still, our results seem to support the intuiton that if uncertainty estimates on the model parameters are not available, constraining policy updates results in improved performance, even though learning-based control algorithms that leverage uncertainty estimates on the model parameters are tough to outperform. In future work, we could consider robust training of the prediction models to account for the covariate shift induced by policy iteration, and sharpening the bounds we developed by leveraging more problem specific structure, like smoothness of the dynamics.

9 Appendix

Code can be found here: https://tinyurl.com/rrr-cs329d

References

[1] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[2] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. 2020.

[3] Simon Schaefer, Karen Leung, Boris Ivanovic, and Marco Pavone. Leveraging neural network gradients within trajectory optimization for proactive human-robot interactions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9673–9679, 2021.

[4] Bandyopadhyay T., Won K. S., Frazzoli E., Hsu D., Lee W. S., and Rus D. Intention-aware motion planning. *In: Frazzoli E., Lozano-Perez T., Roy N., Rus D. (eds) Algorithmic Foundations of Robotics X. Springer Tracts in Advanced Robotics, vol*, 86., 2013.

[5] Riccardo Spica, Davide Falanga, Eric Cristofalo, Eduardo Montijano, Davide Scaramuzza, and Mac Schwager. A Real-Time Game Theoretic Planner for Autonomous Two-Player Drone Racing. *IEEE Transactions on Robotics*, 36(5), 2020.

[6] Lukas Brunke. Learning Model Predictive Control for Competitive Autonomous Racing. *arXiv:2005.00826 [cs, math, stat]*, May 2020. arXiv: 2005.00826.

[7] David Fridovich-Keil, Ellis Ratner, Lasse Peters, Anca D. Dragan, and Claire J. Tomlin. Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1475–1481, 2020.

[8] Andrew Y. Ng and Stuart Russell. Algorithms for Inverse Reinforcement Learning. In *in Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, 2000.

[9] Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for Autonomous Cars that Leverage Effects on Human Actions. page 9.

[10] Lukas Hewing, Juraj Kabzan, and Melanie N. Zeilinger. Cautious Model Predictive Control using Gaussian Process Regression. May 2017.

[11] Anirudha Majumdar, Alec Farid, and Anoopkumar Sonar. Pac-bayes control: learning policies that provably generalize to novel environments. *The International Journal of Robotics Research*, 40(2-3):574–593, 2021.

[12] Robert Dyro, James Harrison, Apoorva Sharma, and Marco Pavone. Particle mpc for uncertain and learning-based control, 2021.

[13] Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 465–472, 2011.

[14] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models, 2018.

[15] Giuseppe C. Calafiore and Lorenzo Fagiano. Robust model predictive control via scenario optimization. *IEEE Transactions on Automatic Control*, 58(1):219–224, 2013.

[16] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017.

[17] Aman Sinha, Matthew O'Kelly, Hongrui Zheng, Rahul Mangharam, John Duchi, and Russ Tedrake. FormulaZero: Distributionally Robust Online Adaptation via Offline Population Synthesis. In *International Conference on Machine Learning*, pages 8992–9004. PMLR, November 2020. ISSN: 2640-3498.

[18] Clara Fannjiang and Jennifer Listgarten. Autofocused oracles for model-based design, 2020.

[19] Carl Edward Rasmussen, Christopher K. I. Williams, and Francis Bach. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[20] Donsker MD and Varadhan SRS. *Asymptotic evaluation of certain Markov process expectations for large time*. Communications on Pure and Applied Mathematics 28:1–47, 1975.

[21] Gray RM. *Entropy and Information Theory*. New York: Springer Science Business Media, 2011.

[22] Hamed Rahimian and Sanjay Mehrotra. Distributionally robust optimization: A review, 2019.

[23] Hongseok Namkoong. Lecture notes for columbia university b9145: Reliable statistical learning, 3030.