Robust Regression for Safe Exploration in Control

Anqi Liu **Guanya Shi** **Soon-Jo Chung** **Anima Anandkumar** **Yisong Yue**
California Institute of Technology, {anqiliu,gshi,sjchung,anima,yyue}@caltech.edu

Abstract

We study the problem of safe learning and exploration in sequential control problems. The goal is to safely collect data samples from an operating environment to learn an optimal controller. A central challenge in this setting is how to quantify uncertainty in order to choose provably-safe actions that allow us to collect useful data and reduce uncertainty, thereby achieving both improved safety and optimality. To address this challenge, we present a deep robust regression model that is trained to directly predict the uncertainty bounds for safe exploration. We then show how to integrate our robust regression approach with model-based control methods by learning a dynamic model with robustness bounds. We derive generalization bounds under domain shifts for learning and connect them with safety and stability bounds in control. We demonstrate empirically that our robust regression approach can outperform conventional Gaussian process (GP) based safe exploration in settings where it is difficult to specify a good GP prior.

1 Introduction

In many sequential learning tasks, we often must iteratively collect new data and then learn from them, creating a dependency between learning and experiment design. For example, we can learn from previous experimental outcomes to choose the next experiment to optimize future experimental outcomes [1]. This iterative interaction between learning and experiment design poses challenges, most notably how the updated model should inform new data collection. This challenge is further complicated when operating within a dynamical system, as both learning and experiment design must be integrated with dynamics modeling and controller design. We are further interested in the settings that require safety and stability guarantees of the closed-loop controller.

Motivating Application. Consider a motivating problem of safely landing a drone at fast landing speeds (e.g., beyond a human expert's piloting abilities). We typically only have partial knowledge of the underlying dynamics and thus need to collect data to learn a better dynamics model. However, collecting relevant training data poses safety risks as one needs to guarantee that landing quickly will not crash the drone before we have collected such data. We typically start with nominal controller designed based on a nominal dynamics model that has high uncertainty when flying close to the ground at high speeds. Figure 1 describes the setting, where the goal is to eventually learn to execute the orange trajectory while not being overconfident and execute the green trajectory instead (which crashes into the ground); the initial nominal controller may only be able to execute the blue trajectory. The goal then is to iteratively (i.e., episodically) execute increasingly aggressive trajectories, while certifying (with high probability) the safety each trajectory to be executed.

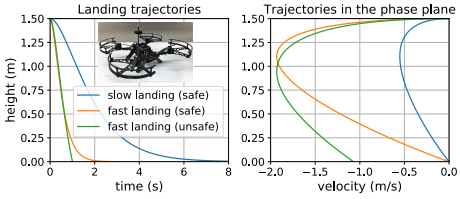

Figure 1: Illustration of Motivation

To date, the most popular methods for safe exploration in dynamical systems are based on Gaussian processes (GPs) [2], mainly due to their straightforward uncertainty quantification mechanism [3, 4, 5, 6, 7]. However, GPs are sensitive to model selection. Safety constraints can be violated if the

Preprint. Under review.

model (i.e., the kernel) is mis-specified and hyperparameters are not well-tuned. For instance, when equipped with a kernel that is overly optimistic in its ability to extrapolate, GP-based exploration can choose overly aggressive behaviors, which is highly undesirable for safety-critical tasks.

Our Contributions. In this paper, we propose a deep robust regression approach for safe exploration in model-based control. We take the perspective that each episode of exploration can be viewed as a data shift problem, i.e., the "test" data in the proposed exploratory trajectory comes from a shifted distribution compared to the current training set. Our approach learns to quantify uncertainty under such covariate shift, which we then use to learn robust dynamics models to quantify uncertainty of entire trajectories for safe exploration. In the drone example above, we would aim to gradually increase the speed of landing by choosing controls for executing higher speed trajectories, while simultaneously reducing uncertainty of dynamics model.

Our approach builds upon robust linear regression under covariate shifts [8], which we extend to training deep neural networks. The resulting method learns to directly predicts an uncertainty bound via minimizing a relative loss in terms of a base distribution using a minimax approach, where the adversarial player is a worst-case competing probabilistic estimator from source data samples. We analyze learning performance from both generalization and data perturbation perspectives, which show a relation between learning errors on the target data and source prediction variances. We utilize spectral normalized neural networks [9, 10], which guarantees certain stability properties, and derive corresponding bounds in the robust regression framework. Our results on robust regression are of independent interest beyond our application to control.

We integrate our robust regression analysis to address exploration in model-based control, and derive safety and stability bounds for control performance when learning robust dynamics models. We propose a novel safe exploration algorithm that guarantees safety during control deployment, and also derive convergence guarantees to optimal dynamics estimator and controller. We empirically show that our approach outperforms conventional GP-based safe exploration with much less tuning effort in two scenarios: (a) inverted pendulum trajectory tracking; and (b) fast drone landing using an aerodynamics simulation based on real-world flight data [10].

2 Robust Regression

2.1 Robust regression under covariate shifts

Covariate shift refers to the distribution shift caused only by the input variables $P(x)$, but not the conditional output distribution $P(y|x)$. This assumption is valid in many cases, especially in dynamical systems. In our motivating safe landing example, there is a universal "true" aerodynamics model, but we typically only observe training data from a small part of the state space. We now briefly recap the original robust regression model. The method tries to robustly minimize a relative loss function defined as the difference in condition log-loss between an estimator $\hat{P}(y|x)$ and a baseline conditional distribution $P_0(y|x)$ on the target data distribution $P_{\text{trg}}(x)P(y|x)$. This loss essentially measures the amount of expected "surprise" in modeling true data distribution $P_{\text{trg}}(x)P(y|x)$ that comes from $P_{\text{trg}}(x)\hat{P}(y|x)$ instead of $P_{\text{trg}}(x)P_0(y|x)$: relative loss $\mathcal{L} := \mathbb{E}_{P_{\text{trg}}(x)P(y|x)}\left[-\log \frac{\hat{P}(Y|X)}{P_0(Y|X)}\right]$. We constrain the true condition probability $P(y|x)$ to satisfy certain statistical properties Γ of the source data distribution $P_{\text{src}}(x)$: $\Gamma := \{P(y|x)||\mathbb{E}_{P_{\text{src}}(x)P(y|x)}[\Phi(X,Y)] - \mathbf{c}| \leq \lambda\}$, where $\mathbf{c} = \frac{1}{n}\sum_{i=n}^{n}\phi(x_i, y_i)$ is a vector of statistics measured from the source data. We then seek to find the regression model that is robust to the "most surprising" distribution that can arise from covariate shift: $\min_{\hat{P}(y|x)} \max_{P(y|x)\in\Gamma} \mathcal{L}$. The solution of this problem has the parametric form: $\hat{P}(y|x) \propto P_0(y|x)e^{\frac{P_{\text{src}}(x)}{P_{\text{trg}}(x)}\theta^T\Phi(x,y)}$, with parameters obtained maximum condition log likelihood estimation with respect to the target distribution: $\theta = \arg\max_\theta \mathbb{E}_{P_{\text{trg}}(x)P(y|x)}\left[\log \hat{P}_\theta(Y|X)\right]$.

2.2 Robust regression with deep neural networks

To apply the ideas in Section 2.1, we observe that one can directly compute the gradient of the original problem, since it is only associated with source training samples, without regularization: $\text{Obj}(\theta) := \mathbb{E}_{P_{\text{trg}}(x)P(y|x)}\left[\log \hat{P}_\theta(Y|X)\right], \nabla_\theta \text{Obj}(\theta) = \mathbb{E}_{P_{\text{src}}(x)\hat{P}_\theta(y|x)}[\Phi(X,Y)] - \mathbf{c}$.

This sidesteps the need to explicitly evaluate the objective function, which we cannot do without ground truth on the target distribution by leveraging structural properties of the linear minimax problem. But it is a non-standard gradient computation in deep learning. We provide a derivation that is straightforward for implementation using deep learning packages. We also present new theoretical results in Section 2.3.

If we utilize a quadratic feature function on the last layer of the network and incorporate additional parameters $\Theta = [\Theta_x, \Theta_y]$, we obtain a Gaussian distribution and the following form of mean and variance of the predicted output variable:

$$\mu(\phi(x), \Theta) \tag{1}$$
$$= (2r_x\Theta_y + \sigma_0^{-2})^{-1}(-2r_x\Theta_x^T\phi(x) + \mu_0\sigma_0^{-2}),$$
$$\sigma^2(\Theta) = (2r_x\Theta_y + \sigma_0^{-2})^{-1}, \tag{2}$$

Algorithm 1 Stochastic Gradient Descent for Deep Robust Regression under Covariate Shift

Input: Source data points $\{(x_i, y_i)\}$, density ratios $r_i = \frac{P_{\text{src}}(x_i)}{P_{\text{trg}}(x_i)}$, DNN $\phi(x)$ with initialization, DNN SGD optimizer Opt, learning rate γ, regularizer λ, epoch number T.
$\Theta \leftarrow$ random initialization, epoch $= 0$
While epoch $< T$
  For each mini-batch
    Evaluate $\phi(x)$
    Compute $\mu(\phi(x), \Theta)$ with Eq. 1;
    Compute $\sigma(x, \Theta)$ with Eq. 2.
    Compute $\nabla_{\Theta_x}\text{Obj}(\Theta)$ with Eq. 4;
    Compute $\nabla_{\Theta_y}\text{Obj}(\Theta)$ with Eq. 3.
    $\Theta_x \leftarrow \Theta_x - \gamma(\nabla_{\Theta_x}\text{Obj}(\Theta)$;
    $\Theta_y \leftarrow \Theta_y - \gamma(\nabla_{\Theta_y}\text{Obj}(\Theta)$;
    Back-Propagate through networks.
    $Opt(\gamma).step()$
Output: Trained NN $\phi(x)$ and Θ

where (μ_0, σ_0^2) is the mean and variance of the base distribution $P_0(y|x)$, $\phi(x)$ is the output of the deep neural networks, the corresponding quadratic feature function is $\Phi(x, y) = \text{Vector}([y \;\; \phi(x)^T]^T[y \;\; \phi(x)^T])$, and r_x is the density ratio of a data point $\frac{P_{\text{src}}(x)}{P_{\text{trg}}(x)}$. Note that we regard r_x as density ratio estimated independent from this framework. Then the gradient for the additional parameters Θ becomes:

$$\nabla_{\Theta_y}\text{Obj}(\Theta) = \frac{1}{n}\sum_i^n y_i^2 - \frac{1}{n}\sum_i^n \mu^2(\phi(x_i), \Theta) - \sigma^2(\Theta), \tag{3}$$

$$\nabla_{\Theta_x}\text{Obj}(\Theta) = \frac{1}{n}\sum_i^n (y_i - \mu(\phi(x_i), \Theta))\phi(x_i). \tag{4}$$

We then back-propagate the gradients to variables in NN. We show a gradient descent version for learning both the parameters in NN and Θ in Algorithm 1.

Figure 2 shows an intermediate step using robust regression to explore stateless space. Intuitively, the method produces less certain predictions where there is less training data distribution support. The density ratio and base distribution $P_0(y|x)$ determine the uncertainty, and uncertainty is only reduced when there are training data samples. Moreover, when the mean estimator is inaccurate, variance estimator is large to compensate the possible error in prediction with larger uncertainty. We refer a more detailed discussion of the stateless case to the Appendix A.2.6.

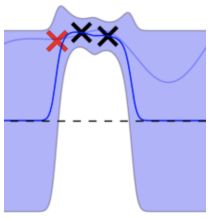

Figure 2: Robust Regression with 3 data points

2.3 Learning performance analysis

The learning performance of our deep robust regression approach can be analyzed from two perspectives: generalization error under distribution shift and error bound under data perturbation based on Lipschitz continuity. We first establish a general form of the bounds and then derive concrete versions for both linear and deep predictors. The proofs are in the appendix.

Theorem 1. [Generic generalization and perturbation bounds] *Assume S is a training set with i.i.d. data $x_i, ..., x_n$ sampled from $P_{src}(x)$, $\mathcal{F}$ is a regression function class satisfying $\sup_{x\in\mathcal{X}, f, f'\in\mathcal{F}} |f(x) - f'(x)| \leq M$, $\hat{\mathfrak{R}}_S(\mathcal{F})$ is the Rademacher complexity on S, W is the upper bound of true density ratio $\sup_{x\sim P_{src}(x)} \frac{P_{trg}(x)}{P_{src}(x)} \leq W$, $\theta_y \in (0, B]$, the weight estimation $\sup_{x\in S} r(x) \leq R$, base distribution variance is σ_0^2, and λ is the upperbound of all λ_i among the dimensions of $\phi(x)$. When learning a $\hat{f} \in \mathcal{F}$ on $P_{trg}(x, y)$, the following generalization error*

bound holds with probability at least $1-\delta$,

$$\mathbb{E}_{P_{trg}(\mathbf{x},y)}[(y-\hat{f}(x))^2] \leq W\left[(2RB+\sigma_0^{-2})^{-1}+\lambda+4M\hat{\mathfrak{R}}_S(\mathcal{F})+3M^2\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right].$$

If we assume that target data samples x*'s stay in a ball* $\mathbb{B}(\epsilon)$ *with diameter* ϵ *from the source data* S, $\mathbb{B}(\epsilon)=x|\sup_{x'\in S}\parallel x-x'\parallel \leq \epsilon$, *the true function* $f(x)$ *is Lipschitz continuous with constant* L, *and the robust regression mean estimator* $\hat{f}$ *is also Lipschitz continuous with constant* $\hat{L}$, *then,*

$$\sup_{x\in\mathbb{B}(\epsilon),y\sim f(x)}[(y-\hat{f}(x))^2] \leq ((2RB+\sigma_0^{-2})^{-1/2}+\sqrt{\lambda}+\left(L+\hat{L}\right)\|\epsilon\|)^2. \tag{5}$$

We can further upper bound the Rademacher complexity if we know the function class.

Corollary 1. [The linear case] *If* $\mathcal{H}$ *is linear function class with* $\parallel\theta_x\parallel \leq A$, *i.e. we only use linear features for* $\phi(x)$, *and* $\sup_{x\in S}\parallel x\parallel \leq \mathfrak{X}$, *the following holds with probability at least* $1-\delta$,

$$\mathbb{E}_{P_{trg}(x,y)}[(y-\hat{f}(x))^2] \leq W\left[(2RB+\sigma_0^{-2})^{-1}+\lambda+\frac{8A^2\mathfrak{X}^2}{B^2}+\frac{3A^2\mathfrak{X}^2}{B^2}\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right].$$

The corresponding perturbation bounds for a linear function class is,

$$\sup_{x\in\mathbb{B}(\epsilon),y\sim f(x)}(y-\hat{f}(x))^2 \leq ((2RB+\sigma_0^{-2})^{-1/2}+\sqrt{\lambda})+\left(L+\frac{A}{B}\right)||\epsilon||)^2. \tag{6}$$

For deep robust regression, we utilize spectral normalized deep neural networks [9] and upper bound the Radermacher Complexity using the bounded spectral complexity [9].

Corollary 2. [The neural network case] *Let neural networks* $\phi(x)=F_{\mathcal{A}}(x)$ *use* L *fixed nonlinearities* $(\sigma_1,...,\sigma_L)$, *which are* ρ_i*-Lipschitz and* $\sigma_i(0)=0$. *Let reference matrices* $(C_1,..,C_L)$ *be given, as well as spectral norm bounds* $(s_i)_{i=1}^L$, *and* l_1 *norm bounds* $(b_i)_{i=1}^L$. *If* $\sqrt{\sum_i||x_i||_2^2}\leq I$, *for robust regression using network* $F_{\mathcal{A}}$ *with weight matrices* $\mathcal{A}=(A_1,...,A_L)$ *and maximum dimension of each layer is at most* D *obey* $||A_i||_\sigma \leq s_i$, $||A_i^T-C_i^T||_{2,1}\leq b_i$, *and* $||F_{\mathcal{A}}(x)||_2\leq\mathfrak{X}$, *the following holds with probability at least* $1-\delta$,

$$\begin{aligned}&\mathbb{E}_{P_{trg}(x,y)}[(y-\hat{f}(x))^2]\\ &\leq W\left[(2RB+\sigma_0^{-2})^{-1}+\lambda+\frac{32A\mathfrak{X}}{Bn^{\frac{3}{2}}}+\frac{288A^2\mathfrak{X}}{nB^2I}\ln n\sqrt{\mathcal{R}_{\mathcal{A}}\ln(2D^2)}+\frac{3A^2\mathfrak{X}^2}{B^2}\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right],\end{aligned} \tag{7}$$

where $\mathcal{R}_{\mathcal{A}}$ *is the spectral complexity of networks* $F_{\mathcal{A}}(x)$, $\mathcal{R}_{\mathcal{A}}:=\left(\prod_{j=1}^L s_j^2\rho_j^2\right)\left(\sum_{i=1}^L(\frac{b_i}{s_i})^{\frac{2}{3}}\right)^3$; *The corresponding perturbation bounds for spectral normalized deep neural networks is,*

$$\sup_{x\in\mathbb{B}(\epsilon),y\sim f(x)}[(y-\hat{f}(x))^2] \leq ((2RB+\sigma_0^{-2})^{-1/2}+\sqrt{\lambda})+\left(L+\frac{A}{B}\mathcal{R}_{\mathcal{A}}\right)||\epsilon||)^2 \tag{8}$$

3 Control with a Robust Regression Dynamics Estimator

We propose to learn the non-linear dynamics using deep robust regression under covariate shift, which is used to safely explore and collect data to improve model accuracy and derive improved control polices.[1] In order to connect learning to control, we need to adapt the general analysis of robust regression to fit the control context, i.e., to analyze entire trajectory behaviors. Instead of assuming the target data is IID samples from a static target data distribution, which can be too conservative considering robustness on a large target data distribution consisting of all possible unseen states, we propose to only consider one trajectory as the targe data distribution at a time, and to use robust dynamics regression to establish safety and stability guarantees. We then enlarge the set of safe trajectories episodically by collect training data along the executed trajectory and update the dynamics model accordingly. We now introduce the dynamics model and controller design. Note that all the norms in this section are the 2-norm. All proofs are in the appendix.

[1] Note that robust regression is also applicable to the standard experimental design setting (e.g., Bayesian optimization), and we conduct an empirical evaluation of that setting in the Appendix A.2.6.

3.1 A mixed model for robotic dynamics

Consider the following mixed model for continuous robotic (drone & pendulum in our experiments) dynamics:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) - Bu = \underbrace{d(q,\dot{q})}_{\text{unknown}}, \tag{9}$$

with generalized coordinates $q \in \mathbb{R}^n$ (and their first & second time derivatives, $\dot{q}$ & $\ddot{q}$), control input $u \in \mathbb{R}^m$, inertia matrix $M(q) \in \mathbb{S}^n_{++}$, centrifugal and Coriolis terms $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$, gravitational forces $G \in \mathbb{R}^n$, actuation matrix $B \in \mathbb{R}^{n\times m}$ and some unknown residual dynamics $d \in \mathbb{R}^n$. Note that the C matrix is chosen to make $\dot{M} - 2C$ skew-symmetric from the relationship between the Riemannian metric $M(q)$ and Christoffel symbols. Here d is general, which potentially captures both parametric and nonparametric unmodelled terms.

Definition of safety in trajectory tracking. The state vector is denoted as $x(t) = [q(t), \dot{q}(t)]$, $x \in \mathbb{R}^{2n}$. The main objective for a robotic system is to track some time-varying desired trajectory $x_d(t) = [q_d(t), \dot{q}_d(t)]$. Simultaneously, we want to guarantee safety: $x(t) \in \mathfrak{S}, \forall t$, with high probability, where $\mathfrak{S}$ is some safety set. It is obvious that $x_d(t) \in \mathfrak{S}, \forall t$. However, because we do not know d a priori, the tracking error $\tilde{x}(t) \triangleq x(t) - x_d(t)$ may be large such that $\exists t, x(t) \notin \mathfrak{S}$. There are two 1-d examples following.

Example 1 (inverted pendulum with external wind). In addition to the classical pendulum model, we consider some unknown external wind. Dynamics can be described as $ml^2\ddot{\alpha} - mlg\sin\alpha - u = d(\alpha,\dot{\alpha})$, where $d(\alpha,\dot{\alpha})$ is external torque generated by the unknown wind. Our control goal is to track $\alpha_d(t) = \sin(t)$, and the safety set is $\mathfrak{S} = \{(\alpha,\dot{\alpha}) : |\alpha| < 1.5\}$.

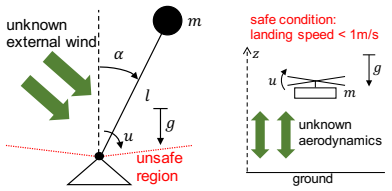

Figure 3: Illustration of two examples

Example 2 (drone landing with ground effect) For this example, we consider drone landing with unknown ground effect. Dynamics is $m\ddot{z} + mg - c_T u^2 = d(z,\dot{z})$, where c_T is the thrust coefficient. The control goal is smooth and quick landing, i.e., quickly driving $z \to 0$. The safety set is $\mathfrak{S} = \{(z,\dot{z}) : \text{when } z = 0, \dot{z} > -1\}$, i.e., the drone cannot hit the ground with high velocity.

3.2 Model based nonlinear control

We design a nonlinear controller, which can leverage robust regression of $d(q,\dot{q})$ in Eq. 9. Define the reference trajectory as $\dot{q}_r = \dot{q}_d - \Lambda\tilde{q}$, where $\tilde{q} = q - q_d$, and the composite variable as $s = \dot{q} - \dot{q}_r = \dot{\tilde{q}} + \Lambda\tilde{q}$, where Λ is positive definite. The control objective is to drive s to 0 or a small error ball in the presence of bounded uncertainty. With the robust estimation of $d(q,\dot{q})$, $\hat{d}(q,\dot{q})$, we propose the following nonlinear controller:

$$u = B^\dagger(M(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r - Ks + G(q) - \hat{d}(q,\dot{q})), \tag{10}$$

where K is a positive definite matrix, and $\dagger$ denotes the Moore-Penrose pseudoinverse.

With the control law Eq. 10, we will have the following closed-loop dynamics:

$$\begin{bmatrix} M(q) & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} \dot{s} \\ \dot{\tilde{q}} \end{bmatrix} + \begin{bmatrix} C(q,\dot{q}) + K & 0 \\ -I & \Lambda \end{bmatrix}\begin{bmatrix} s \\ \tilde{q} \end{bmatrix} = \begin{bmatrix} d - \hat{d} \\ 0 \end{bmatrix} = \begin{bmatrix} \epsilon \\ 0 \end{bmatrix}. \tag{11}$$

where $\epsilon = d - \hat{d}$ is the approximation error between d and $\hat{d}$.

3.3 Stability analysis and trajectory bound

To connect learning with control, we set $\|\epsilon\|^2 = \|d - \hat{d}\|^2$ to correspond with the bounds in Section 2.3. The first option is to connect to Theorem 1, where target data is only a single trajectory $x_{\text{trg}}(t)$ that deviates from those in the training data, which means W is further bounded. The second option is to use a perturbation bound, where $x_{\text{trg}}(t) \in \mathbb{B}(\epsilon)$. We omit rewriting of the bounds here and refer to appendix A.1.1, but emphasize that $\|\epsilon\|$ is upper bounded with $\|\epsilon\| \leq \sup_{x\in x_{\text{trg}}(t)} \|\epsilon(x)\|$ when

we define target data in a specific set and use robust regression for learning dynamics. We show $\|x(t) - x_d(t)\| \triangleq \|\tilde{x}(t)\|$ (Euclidean distance between the desired trajectory and the real trajectory) is bounded when the error of the dynamics estimation is bounded. Again, recall that $x = [q, \dot{q}]$ is our state, and $x_d(t)$ is the desired trajectory.

Theorem 2. *Suppose x is in some compact set $\mathcal{X}$, and $\epsilon_m = \sup_{x\in\mathcal{X}} \|\epsilon\|$. Then $\tilde{x}$ will exponentially converge to the following ball: $\lim_{t\to\infty} \|\tilde{x}(t)\| = \gamma \cdot \epsilon_m$, where*

$$\gamma = \frac{\lambda_{\max}(M)}{\lambda_{\min}(K)\lambda_{\min}(M)} \sqrt{(\frac{1}{\lambda_{\min}(\Lambda)})^2 + (1 + \frac{\lambda_{\max}(\Lambda)}{\lambda_{\min}(\Lambda)})^2}. \tag{12}$$

3.4 Safe exploration algorithm

Theorem 2 indicates that if we can design a compact set $\mathcal{X}$ and find the corresponding maximum error bound $\epsilon_m = \sup_{x\in\mathcal{X}} \|\epsilon\|$ on it, we can use it to decide whether a trajectory in this set is safe or not by checking whether its worst-case possible tracking trajectory is in the safety set $\mathfrak{S}$.

In practice, we design a pool of desired trajectories and use the current predictor of the dynamics to find the worst-case possible tracking trajectory for each of the desired ones. Note that the worst-case tracking trajectories can be computed by generating a "tube" using euclidean distance in Theorem 2. We then eliminate unsafe ones and choose the most "aggressive" one in terms of the primary objective function for the next iteration. Instead of evaluating the actual upper bound, we use $\beta \max_x \sigma(x)$ for measuring ϵ_m as an approximation, since it is guaranteed that the error is within $\beta \max_x \sigma(x)$ with high probability as long as the prediction is a Gaussian distribution, if the true function is drawn from the same distribution. We refer a detailed discussion to Appendix A.1.2. Algorithm 2 describes this safe exploration procedure.

Algorithm 2 Safe Exploration for Control using Robust Dynamics Estimation

Input: Pool of desired trajectories with parameter k, $x_d^k(t), k = 1, 2, ..., K$, cost function J, robust regression model of dynamics f, controller U, safety set $\mathfrak{S}$, base distribution $\mathcal{N}(\mu_0, \sigma_0^2)$, parameter β
Dynamics model $f_0 = \mathcal{N}(\mu_0, \sigma_0^2)$
Training set = $\emptyset$, $f = f_0$
While $t = 1, ..., T$
  Safe trajectory list $L = \emptyset$
  For $i = 1, ..., K$
    Predict $(\mu, \sigma^2) = f(x_d^k(t))$
    $\sigma_m = \max \sigma(x_d^k(t))$; $\epsilon_m = \beta\sigma_m$
    If worst-case trajectory in $\mathfrak{S}$
      Add $x_d^k(t)$ to L
  Track $x_d^*(t) = \arg\min_{x_d(t)\in L} J(x_d(t))$
  to collect data $x'(t)$ using controller U
  Add data $x'(t)$ to Training set
  Train dynamics model f', $f = f'$
Output: dynamics model f, last desired trajectory $x_T(t)$ and actual trajectory $x'_T(t)$

3.5 Convergence analysis

We show that using Algorithm 2, we are able to reach optimality in terms of learning the dynamics model, i.e. converge to the optimal predictor in the function class.

Theorem 3. *If there exists an optimal predictor f^* in function class $\mathcal{F}$, $f^* = \arg\min_{f\in\mathcal{F}, x\in\mathcal{X}} (y - f(x))^2$, with mean and variance estimates $(\mu^*(x), \sigma^2(x)^*)$, the sequence of estimator from robust regression in each step consist of $\mu_0, \mu_1, ..., \mu_t$ and $\sigma_0, \sigma_1, ..., \sigma_t$, for any ϵ, the output of Algorithm 2 converge to f^*, $\|\hat{f}(x) - f^*(x)\| \leq \epsilon$ with T as the smallest integer that satisfies the following with at least probability $1 - \delta$:*

$$\sum_{t=1}^{T} \sum_{i=1}^{t} \sqrt{2(\sigma_i^2(x) + \sigma_{i-1}^2(x)) \log\left(\frac{|\mathcal{X}|}{2\delta}\right)} + (\mu_t(x) - \mu^*(x)) \leq T\epsilon. \tag{13}$$

Given the convergence of the dynamics model, we can prove the optimal desired trajectory is collected and tracked with good control performance at the end.

Corollary 3. *If there exist an optimal trajectory parameter k^* for the controller to track $x_d^*(t)$ safely and obtain minimal cost function among all the safe trajectories when the estimated dynamic model $\hat{f}(x)$ satisfies $\|\hat{f}(x) - f^*(x)\| \leq \epsilon$ for all $x \in \mathcal{X}$, $x_d^*(t)$ is collected by Algorithm 2, as well as being tracked with $\|x'(t) - x_d^*(t)\| \leq \gamma\epsilon$ for all t, with γ in Theorem 2, where $x'(t)$ is the tracking trajectory using $f^*(x)$ as dynamics estimation.*

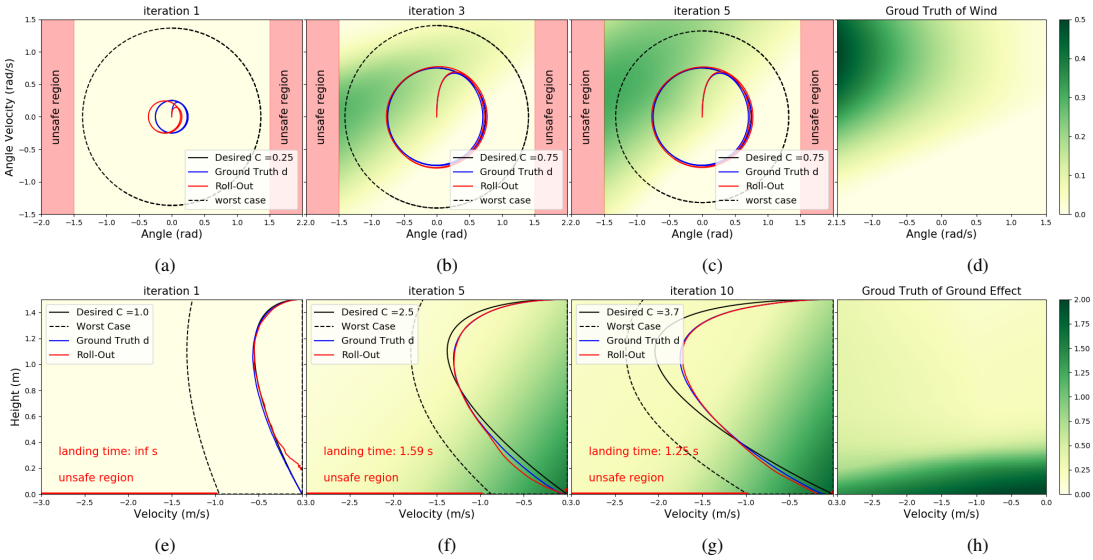

Figure 4: **Top Row.** Results on the pendulum task: (a) the first, (b) the third, and (c) the fifth iteration phase portrait of angle and angular velocity, dashed line shows the worst-case possible trajectory in tracking, according to Theorem 2; heatmap shows the prediction of unknown dynamics (the wind); (d) the unknown dynamics ground truth. **Bottom Row.** Results on the drone landing task: (e) the first, (f) the fifth, and (g) the tenth iteration phase portrait with height and velocity; heatmap shows the prediction of unknown dynamics (the ground effect); (h) the ground effect ground truth.

4 Experiments

We conduct experiments on simulation on the inverted pendulum and drone landing examples as discussed in Section 3.1. We use kernel density estimation to estimate density ratios. We demonstrate that our approach can reliably and safely converge to optimal behavior. We also compare with a Gaussian process (GP) version of Algorithm 2. In general, we find that it is difficult to tune kernel parameters, and all GP models underperform compared to our approach.

Example 1 (inverted pendulum with external wind). Recall that the safety set is $\mathfrak{S} = \{(\alpha, \dot{\alpha}) : |\alpha| < 1.5\}$ in the pendulum case, and the final control goal is to track $\alpha_d(t) = \sin(t)$. Therefore our desired trajectory pool is $\mathcal{P}(C) = \{\alpha_d(t) = C \cdot \sin(t), 0 < C \leq 1\}$. The ground truth of wind comes from quadratic air drag model. We use the angle upper bound in trajectory as the reward function for choosing "most aggressive" trajectories. Figure 4 demonstrates the exploration process with selected desired trajectories, worst-case tracking trajectory under current dynamics model, tracking trajectories with the ground truth unknown dynamics model, and actual tracking trajectories. Here we use base distribution $\mathcal{N}(0, 0.4)$ to start with and $\beta = 0.5$. As shown in Figure 4 (a) to (c), the algorithm selects small C to guarantee safety at the beginning, and gradually is able to select larger C values and track it with small error.

Example 2 (drone landing with ground effect) Recall that the safety set is $\mathfrak{S} = \{(z, \dot{z}) :$ when $z = 0, \dot{z} > -1\}$, which means the drone can not hit the ground with high velocity. Our desired trajectory pool is $\mathcal{P}(C, h_d) = \{z_d(t) = e^{-Ct}(1+Ct)(1.5-h_d)+h_d, 0 < C, 0 \leq h_d < 1.5\}$, which means the drone smoothly moves from $z(0) = 1.5$ to the desired height h_d. If $h_d = 0$, the drone lands successfully. Note that greater C means faster landing. We use smaller landing time as the reward function that determines the next "aggressive" trajectory. The ground truth of aerodynamics in landing comes from a dynamics simulator that is trained in [10], where $d(z, \dot{z})$ is a four-layer ReLU neural network trained by real flying data. Here we use base distribution $\mathcal{N}(0, 1)$ for robust regression and $\beta = 1$. Results in Figure 4(e) to (g) demonstrate that, because of the lack of aerodynamics $d(z, \dot{z})$, $h_d = 0$ and big C may not be safe at the beginning. Starting from conservative desired trajectories, the safe exploration using robust regression is able to track more aggressive desired trajectory with $h_d = 0$ and big C while staying safe.

Comparison with GPs. We examine here drone landing time, and defer examining the simpler pendulum setting to the appendix. We compare against five GP models with a wide range of kernel parameters, including both ones that are optimistic or conservative about their prediction uncertainty, by setting different bandwidths in the RBF kernel. Figure 5 shows that our approach outperforms all GP models. Modeling the ground effect is notoriously challenging [10], and the GP suffers from model mis-specification. In contrast, our approach can fit general non-linear function estimators such as deep neural networks adaptively to the available data, which leads to more flexible inductive bias and better fitting of the data and uncertainty quantification. Additional results for the drone landing setting as well as inverted pendulum are in Appendix A.5. The supplemental material also contains a video demoing the results.

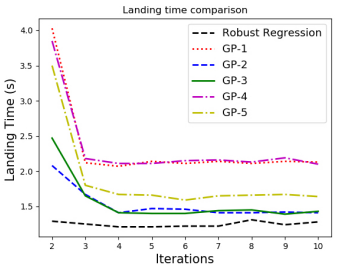

Figure 5: Comparison with GPs

5 Related Work

Safe Exploration. Safe exploration methods commonly use Gaussian processes (GPs) to quantify uncertainty [11, 12, 13, 3, 4, 14, 15, 5, 6, 7]. These methods are related to bandit algorithms [16] and typically employ upper confidence bounds [17] to balance exploration versus exploitation [18]. As discussed above, GP-based approaches can be sensitive to model selection. One could blend GP-based modeling with general function approximations (such as deep learning) [5, 19], but the resulting optimization-based control problem can be challenging to solve. Other approaches either require having a safety model pre-specified upfront [20], are restricted to relatively simple models [21], have no convergence guarantees during learning [22], or have no guarantees at all [23]. Our work also shares some similarity with [24, 25], which use deep neural networks for sampling-based optimization; however, those approaches have no guarantees and so are unsuitable for safe exploration.

Distribution Shift. The study of data shift has seen increasing interest in recent years, owing to the widespread practical issue that real test distributions rarely match the training distribution. Our work is stylistically similar to [26, 8, 27, 28], which also frame uncertainty quantification through the lens of covariate shift, although ours is the first to extend to deep neural networks with rigorous guarantees. More broadly, dealing with domain shift is a fundamental challenge in deep learning, as highlighted by their vulnerability to adversarial inputs [29], and the implied lack of robustness. Beyond robust estimation, the typical approaches are to either regularize [30, 31, 32, 9, 33, 10, 34, 35] or synthesize an augmented dataset that anticipates the domain shift [36, 37, 38]. We take the former approach by employing spectral normalization [9, 10] in conjunction with robust estimation.

6 Conclusion & Future Work

In this paper, we propose an algorithmic framework for safe exploration in model-based control. To quantify uncertainty, we develop a robust deep regression method for dynamics estimation. Using robust regression, we explicitly deal with data shifts during episodic learning, and in particular can quantify uncertainty over entire trajectories. We prove the generalization and perturbation bounds for robust regression, and show how to integrate with control to derive safety bounds in terms of stability. These bounds explicitly translates the error in dynamics learning to the tracking error in control. From this, we design a safe exploration algorithm based on a finite pool of desired trajectories. We prove that the proposed safe exploration algorithm converges to the optimal dynamics estimator in its function class, as well as the optimal controller for tracking optimal desired trajectories.

There are many avenues for future work. For instance, our safety criterion was relatively simple, and one can consider employing more sophisticated criteria that require more sophisticated certification approaches such as reachability analysis [6]. Our theoretical analysis can also be improved, since tighter safety guarantees can lead to dramatically improved performance. Directions to explore include incorporating other regularization techniques, relaxing from Gaussian observation noise, and incorporating more sophisticated density estimation techniques. Another interesting direction is to incorporate with deep kernel learning [39] for safe exploration, which does make stronger assumptions (uses a Gaussian process model) but might alleviate some issues with kernel parameter tuning. Finally, our deep robust regression approach is of independent interest beyond model-based control, and can be incorporated in other applications as well.

References

[1] Ruben Martinez-Cantin. Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits. *The Journal of Machine Learning Research*, 15(1):3735–3739, 2014.

[2] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[3] Anayo K Akametalu, Jaime F Fisac, Jeremy H Gillula, Shahab Kaynama, Melanie N Zeilinger, and Claire J Tomlin. Reachability-based safe learning with gaussian processes. In *53rd IEEE Conference on Decision and Control*, pages 1424–1431. IEEE, 2014.

[4] Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 491–496. IEEE, 2016.

[5] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems*, pages 908–918, 2017.

[6] Jaime F Fisac, Anayo K Akametalu, Melanie N Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 2018.

[7] Hassan Khalil and Jessy Grizzle. Nonlinear systems. *Prentice hall*, 2002.

[8] Xiangli Chen, Mathew Monfort, Anqi Liu, and Brian D Ziebart. Robust covariate shift regression. In *Artificial Intelligence and Statistics*, pages 1270–1279, 2016.

[9] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

[10] Guanya Shi, Xichen Shi, Michael O'Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. *International Conference on Robotics and Automation (ICRA)*, 2019.

[11] Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. Safe exploration for optimization with gaussian processes. In *International Conference on Machine Learning*, pages 997–1005, 2015.

[12] Yanan Sui, Vincent Zhuang, Joel W Burdick, and Yisong Yue. Stagewise safe bayesian optimization with gaussian processes. In *International Conference on Machine Learning (ICML)*, 2018.

[13] Johannes Kirschner, Mojmír Mutnỳ, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. Adaptive and safe bayesian optimization in high dimensions via one-dimensional subspaces. In *International Conference on Machine Learning (ICML)*, 2019.

[14] Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration in finite markov decision processes with gaussian processes. In *Advances in Neural Information Processing Systems*, pages 4312–4320, 2016.

[15] Akifumi Wachi, Yanan Sui, Yisong Yue, and Masahiro Ono. Safe exploration and optimization of constrained mdps using gaussian processes. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[16] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[17] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[18] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning (ICML)*, 2010.

[19] Richard Cheng, Gábor Orosz, Richard M Murray, and Joel W Burdick. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In *Conference on Artificial Intelligence (AAAI)*, 2019.

[20] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[21] Teodor Mihai Moldovan and Pieter Abbeel. Safe exploration in markov decision processes. In *International Conference on Machine Learning (ICML)*, 2012.

[22] Andrew J Taylor, Victor D Dorobantu, Hoang M Le, Yisong Yue, and Aaron D Ames. Episodic learning with control lyapunov functions for uncertain robotic systems. *arXiv preprint arXiv:1903.01577*, 2019.

[23] Javier Garcia and Fernando Fernández. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 45:515–564, 2012.

[24] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180, 2015.

[25] David H Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *International Conference on Machine Learning (ICML)*, 2019.

[26] Anqi Liu, Lev Reyzin, and Brian D Ziebart. Shift-pessimistic active learning using robust bias-aware prediction. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[27] Anqi Liu and Brian Ziebart. Robust classification under sample selection bias. In *Advances in neural information processing systems*, pages 37–45, 2014.

[28] Anqi Liu and Brian D Ziebart. Robust covariate shift prediction with general losses and feature views. *arXiv preprint arXiv:1712.10043*, 2017.

[29] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[31] Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pages 351–359, 2013.

[32] Hoang M. Le, Andrew Kang, Yisong Yue, and Peter Carr. Smooth imitation learning for online sequence prediction. In *International Conference on Machine Learning (ICML)*, 2016.

[33] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[34] Ari S Benjamin, David Rolnick, and Konrad Kording. Measuring and regularizing networks in function space. In *International Conference on Learning Representations (ICLR)*, 2019.

[35] Richard Cheng, Abhinav Verma, Gabor Orosz, Swarat Chaudhuri, Yisong Yue, and Joel Burdick. Control regularization for reduced variance reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2019.

[36] Alessandro Prest, Christian Leistner, Javier Civera, Cordelia Schmid, and Vittorio Ferrari. Learning object class detectors from weakly annotated video. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3282–3289. IEEE, 2012.

[37] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4480–4488, 2016.

[38] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[39] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.

A Appendix

A.1 Additional Theoretical Results

A.1.1 Improved Bounds for Control

As explained in the paper, we can further improve the learning bounds in the control context when we control the target data in a strategically way. In Theorem 1, W is the upper bound of the true density ratio of this two distribution, which potentially can be very large when target data is a very different one from the source. However, we can choose our next trajectory as the one not deviate too much from the source data in practice, so that further constraining W and also ϵ in Theorem 1. We can rewrite the theorem as:

Theorem 4. [Improved Generalization and perturbation bounds in general cases] *Assume S is a training set S with i.i.d. data $x_i, ..., x_n$ sampled from $P_{src}(x)$, $\mathcal{F}$ is the function class of mean estimator $\hat{f}$ in robust regression, it satisfies $\sup_{x\in\mathcal{X}, f, f'\in\mathcal{F}} |f(x) - f'(x)| \leq M$, $\hat{\mathfrak{R}}_S(\mathcal{F})$ is the Rademacher complexity on S, W is the upper bound of true density ratio $\sup_{x\sim P_{src}(x)} \frac{P_{trg}(x)}{P_{src}(x)} \leq W'$, $\theta_y \in (0, B]$, the weight estimation $\sup_{x\in S} r(x) \leq R$, base distribution variance is σ_0^2, λ is the upperbound of all λ_i among the dimensions of $\phi(x)$, we have the generalization error bound on $P_{trg}(x, y)$ hold with probability $1 - \delta$,*

$$\mathbb{E}_{P_{trg}(\mathbf{x},y)}[(y - \hat{f}(x))^2] \leq W' \left[(2RB + \sigma_0^{-2})^{-1} + \lambda + 4M\hat{\mathfrak{R}}_S(\mathcal{F}) + 3M^2\sqrt{\frac{\log\frac{2}{\delta}}{2n}} \right]$$

If we assume target data samples x's stay in a ball $\mathbb{B}(\epsilon)$ with diameter ϵ' from the source data S, $\mathbb{B}(\epsilon) = x|\sup_{x'\in S} \|x - x'\| \leq \epsilon'$ the true function $f(x)$ is Lipschitz continuous with constant L and the robust regression mean estimator $\hat{f}$ is also Lipschitz continuous with constant $\hat{L}$,

$$\sup_{x\in\mathbb{B}(\epsilon), y\sim f(x)} [(y - \hat{f}(x))^2] \leq ((2RB + \sigma_0^{-2})^{-1/2} + \sqrt{\lambda} + \left(L + \hat{L}\right) \|\epsilon'\|)^2 \tag{14}$$

Note that in generalization bound, we can further improve the bound if we know what is the method for estimating density ratio r and further relate the overall learning performance with the density ration estimation. Here, we just use r as if it is a value that is given to us beforehand.

A.1.2 High Probability Bounds for Gaussian Distribution

In Algorithm 2, we use $\beta\sigma(x)$ as our approximation of the learning error from the robust regression instead of measuring the actual learning upper bound, which is hard to evaluate. Here we give the justification.

If the prediction from robust regression is $\mathcal{N}(\mu(x), \sigma^2(x))$, assuming true function is drawn from the same distribution, we have $Pr\{|f(x) - \mu(x)| > \sqrt{\beta}\sigma(x)\} \leq e^{-\beta/2}$. Also, for a unit normal distribution $r \sim \mathcal{N}(0, 1)$, we have $Pr\{r > c\} = e^{-c^2/2}(2\pi)^{-1/2} \int e^{-(r-c)^2/2 - c(r-c)} dr \leq e^{-c^2/2} Pr\{r > 0\} = (1/2)e^{-c^2/2}$. Therefore, for data S, $|f(x) - \mu(x)| \leq \beta^{-1/2}\sigma(x)$ hold with probability greater than $1 - |S|e^{-\beta\sigma/2}$. Therefore, we can choose β in practice and it corresponds with different probability in bounds.

A.2 Proof of Theoretical Results

A.2.1 Proof of Theorem 1

Proof. We first prove the generalization bound using standard Redemacher Complexity for regression problems:

$$
\begin{aligned}
&\mathbb{E}_{P_{trg}(x,y)}[(y-\hat{y}(x))^2] \\
&= \frac{P_{trg}(x,y)}{P_{src}(x,y)}\mathbb{E}_{P_{src}(x,y)}[(y-\hat{y}(x))^2] && \text{(Covariate Shift Assumption)} \\
&= \frac{P_{trg}(x)}{P_{src}(x)}\mathbb{E}_{P_{src}(x,y)}[(y-\hat{y}(x))^2] \\
&\leq W\left[\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}(x_i))^2 + 4M\hat{\mathfrak{R}}_S(\mathcal{F}) + 3M^2\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right] \\
&= W\left[\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}(x_i))^2 + 4M\hat{\mathfrak{R}}_S(\mathcal{F}) + 3M^2\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right] \\
&\leq W\left[\frac{1}{n}\sum_{i=1}^{n}y_i^2 - \hat{y}(x_i)^2 + 4M\hat{\mathfrak{R}}_S(\mathcal{F}) + 3M^2\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right] \\
&= W\left[\frac{1}{n}\sum_{i=1}^{n}\sigma^2(x_i) + \lambda + 4M\hat{\mathfrak{R}}_S(\mathcal{F}) + 3M^2\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right] && \text{(Gradient of training vanishes:} \\
& && y^2 - (\mu(x)^2 + \sigma^2(x)) - \lambda = 0) \\
&\leq W\left[\frac{1}{n}\sum_{i=1}^{n}\sigma^2(x_i) + \lambda + 4M\hat{\mathfrak{R}}_S(\mathcal{F}) + 3M^2\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right] && (15)
\end{aligned}
$$

where $\sup_{x\in X; f,f'\in\mathcal{F}}|h(x)-h'(x)| \leq M$, $\hat{\mathfrak{R}}_S(\mathcal{F})$ is the Rademacher complexity on the function class of mean estimate Eq. 1, and the variance term $\sigma^2(x)$ is the empirical variance of the robust regression model and follows Eq. 2. This is a data-dependent bound that relies on training samples.

We next prove the perturbation bounds. Assuming x stays in a ball $\mathbb{B}(\delta)$ with diameter δ from the source training data S, $\mathbb{B}(\epsilon) = \{x|\sup_{x'\in S}\|x-x'\| \leq \epsilon\}$, the true function $f(x)$ is Lipschitz continuous with constant L and the mean function of our learned estimator is also Lipschitz continuous with constant $\hat{L}$, then we have

$$
\begin{aligned}
&\sup_{x\in\mathbb{B}(\epsilon), y\sim f(x)}(y-\hat{y}(x))^2 \\
&\leq \sup_{x\in S, y\sim f(x)}(|y-\hat{y}(x)| + (L+\hat{L})\|\epsilon\|)^2 \\
&\leq \sup_{x\in S, y\sim f(x)}(|y-\hat{y}(x)| + (L+\hat{L})\|\epsilon\|)^2 \\
&\leq \sup_{x\in S, y\sim f(x)}(\sqrt{|y-\hat{y}(x)|^2} + (L+\hat{L})\|\epsilon\|)^2 \\
&\leq \sup_{x\in S, y\sim f(x)}(\sqrt{|y^2-\hat{y}^2(x)|} + (L+\hat{L})\|\epsilon\|)^2 \\
&= \sup_{x\in S}(\sqrt{\frac{1}{n}\sum_{i=1}^{n}\sigma^2(x_i) + \lambda} + (L+\hat{L})\|\epsilon\|)^2 && \text{(Gradient of training vanishes:}
\end{aligned}
$$

$$y^2 - (\mu(x)^2 + \sigma^2(x)) - \lambda = 0)$$

$$= \sup_{x \in S}(\sqrt{\frac{1}{n}\sum_{i=1}^{n}\sigma^2(x_i)} + \sqrt{\lambda} + (L + \hat{L})\|\epsilon\|)^2 \tag{16}$$

If we have an upperbound for the parameter $\theta_y \in (0, B]$ and the weight estimation $r \leq R$, we have

$$\frac{1}{n}\sum_{i=1}^{n}\sigma^2(x_i) \leq (2RB + \sigma_0^{-2})^{-1} \tag{17}$$

Therefore, the generalization bound and perturbation bounds can be written as

$$\mathbb{E}_{P_{trg}(x,y)}[(y - \hat{y}(x))^2] \leq W\left[(2RB + \sigma_0^{-2})^{-1} + \lambda + 4M\hat{\mathfrak{R}}_S(\mathcal{F}) + 3M^2\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right] \tag{18}$$

$$\sup_{x \in \mathbb{B}(\epsilon), y \sim f(x)}(y - \hat{y}(x))^2 \leq ((2RB + \sigma_0^{-2})^{-1/2} + \sqrt{\lambda} + (L + \hat{L})\|\epsilon\|)^2 \tag{19}$$

□

A.2.2 Proof of Corollary 1

Proof. If function class is linear, assuming $\|\theta_x\| \leq A$, we can further bound the Rademacher complexity of $h = \mu(x, \Theta)$ when $\mu(x, \Theta)$ is an linear model. Indeed, the mean estimate has the form $w_i \cdot x_i + b$, where $w_i = \frac{-2\theta_x r}{2\theta_y r + \frac{1}{\sigma_0^2}}$ and $b = \frac{\frac{\mu_0}{\sigma_0^2}}{2\theta_y r + \frac{1}{\sigma_0^2}}$. If r is large enough to make $2\theta_y r$ dominating $\frac{1}{\sigma_0^2}$, w_i is approximately $-\frac{\theta_x}{\theta_y}$. When r is very small, w_i is approximately 0. Moreover, in this problem, we usually set $\mu_0 = 0$. Therefore, mean estimate is approximately a linear model $< w_i \cdot x >$ with parameters $\|w_i\|$ upper bounded by $\frac{\|\theta_x\|}{|\theta_y|}$ and lower bounded by 0. Here, we regard r as a weight scalar with no relation of x and $\|w_i\| < \frac{A}{B}$. If we have $\|x\| < \mathfrak{X}$,

$$\begin{aligned}
\hat{\mathfrak{R}}_S(\mathcal{F}) &= \frac{2}{m}\mathbb{E}_\gamma \max_{h \in H}\sum_{i=1}^{m}\gamma_i h(x_i) \\
&= \frac{2}{m}\mathbb{E}_\gamma \max_{\|w_i\| < \frac{A}{B}}\sum_{i=1}^{m}\gamma_i < w_i, x_i > \\
&= \frac{2}{m}\mathbb{E}_\gamma \max_{\|w_i\| < \frac{A}{B}}\sum_{i=i}^{m} < w_i, \gamma_i x_i > \\
&\leq \frac{2}{m}\mathbb{E}_\gamma \max_{\|w_i\| < \frac{A}{B}}\sum_{i=1}^{m}\|w_i\|\|\gamma_i x_i\| \\
&\leq \frac{2A}{mB}\mathbb{E}_\gamma \sum_{i=1}^{m}\|\gamma_i x_i\| \\
&\leq \frac{2A}{mB}\mathbb{E}_\gamma \sum_{i=1}^{m}\|\gamma_i\|\|x_i\| \\
&= \frac{2A\mathfrak{X}}{B}
\end{aligned} \tag{20}$$

Therefore, we have:

$$\mathbb{E}_{P_{trg}(x,y)}[(y - \hat{y}(x))^2] = W\left[(2RB + \sigma_0^{-2})^{-1} + \lambda + \frac{8MA\mathfrak{X}}{B} + 3M^2\sqrt{\frac{\log\frac{1}{\delta}}{2n}}\right]$$

We have $M \leq \frac{A\mathfrak{X}}{B}$,

$$\mathbb{E}_{P_{trg}(x,y)}[(y-\hat{y}(x))^2] = W\left[(2RB+\sigma_0^{-2})^{-1}+\lambda+\frac{8A^2\mathfrak{X}^2}{B^2}+\frac{3A^2\mathfrak{X}^2}{B^2}\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right]$$

For the perturbation, when it is linear, we have $\hat{L} \leq \frac{A}{B}$, therefore it's straightforward that

$$\sup_{x\in\mathbb{B}(\epsilon),y\sim f(x)}(y-\hat{y}(x))^2 \leq ((2RB+\sigma_0^{-2})^{-1/2}+\sqrt{\lambda}+(L+\frac{A}{B})\|\epsilon\|)^2 \tag{21}$$

□

A.2.3 Proof of Corollary 2

The proof heavily adapts results in Lemma A.8 in [9].

Proof. According to Lemma A.8 and Theorem 3.3 in [9], we have for squared loss in regression, the covering number is:

$$\ln\mathcal{N}(\mathcal{L},\epsilon,\|\cdot\|) \leq L_{\mathcal{L}}^2\frac{\|X\|_2^2\ln(2W^2)}{\epsilon^2}\mathcal{R}_{\mathcal{A}}, \tag{22}$$

where $\mathcal{L}$ is the family of things obtained by evaluating X with all choices of the network, which is the function we are investigating, the squared loss on output of a approximate linear layer on top of a neural network in our case, W is the maximum width of the network, X is the data input, ϵ comes from the definition of the cover number, $L_{\mathcal{L}}$ is the Lipschitz constant of loss function, and $\mathcal{R}_{\mathcal{A}}$ is the spectral norm $\mathcal{R}_{\mathcal{A}} := \left(\prod_{j=1}^{L}s_j^2\rho_j^2\right)\left(\sum_{i=1}^{L}(\frac{b_i}{s_i})^{\frac{2}{3}}\right)^3$. In our case, we assume $\|X\|_2^2$ is bounded by $\mathfrak{X}$, $L_{\mathcal{L}}$ is $\frac{2A}{B}$. According to the bound of Radermacher Complexity, if $\ln\mathcal{N}(\mathcal{L},\epsilon,\|\cdot\|)$ is denoted as $\frac{R}{\epsilon^2}$, which means $R = L_{\mathcal{L}}^2\|X\|_2^2\ln(2W^2)\mathcal{R}_{\mathcal{A}}$,

$$\mathcal{L} \leq \inf_{\alpha>0}\left(\frac{4\alpha}{\sqrt{n}}+\frac{12}{n}\int_{\alpha}^{\sqrt{n}}\sqrt{\frac{R}{\epsilon}}d\epsilon\right) = \inf_{\alpha>0}\left(\frac{4\alpha}{\sqrt{n}}+\ln(\sqrt{n}/\alpha)\frac{12\sqrt{R}}{n}\right), \tag{23}$$

where $\inf$ can be optimized at $\alpha=\frac{1}{n}$. This is a general result, regardless of R, which can be different for different loss functions. Therefore, for our case,

$$\mathcal{L} \leq \frac{4}{n^{3/2}}+18\ln n\frac{12R}{n} = \frac{4}{n^{3/2}}+18\ln n\frac{12\sqrt{L_{\mathcal{L}}^2\|X\|_2^2\ln(2W^2)\mathcal{R}_{\mathcal{A}}}}{n} \tag{24}$$

$$= \frac{4}{n^{3/2}}+36\frac{A\mathfrak{X}}{B}\ln n\frac{12\sqrt{\ln(2W^2)\mathcal{R}_{\mathcal{A}}}}{n} \tag{25}$$

Therefore, when we know $M \leq \frac{A\mathfrak{X}}{B}$,

$$\begin{aligned}&\mathbb{E}_{P_{trg}(x,y)}[(y-\hat{f}(x))^2]\\ &\leq W\left[(2RB+\sigma_0^{-2})^{-1}+\lambda+\frac{32A\mathfrak{X}}{Bn^{\frac{3}{2}}}+\frac{288A^2\mathfrak{X}}{nB^2I}\ln n\sqrt{\mathcal{R}_{\mathcal{A}}\ln(2W^2)}+\frac{3A^2\mathfrak{X}^2}{B^2}\sqrt{\frac{\log\frac{2}{\delta}}{2n}}\right],\end{aligned} \tag{26}$$

In the perturbation case, the Lipschitz constant of our spectral normalized network is $\frac{A\mathcal{R}_{\mathcal{A}}}{B}$, we have

$$\sup_{x\in\mathbb{B}(\epsilon),y\sim f(x)}[(y-\hat{f}(x))^2] \leq ((2RB+\sigma_0^{-2})^{-1/2}+\sqrt{\lambda})+\left(L+\frac{A}{B}\mathcal{R}_{\mathcal{A}}\right)||\epsilon||)^2 \tag{27}$$

□

A.2.4 Proof of Theorem 2

Proof. Consider the following Lyapunov function:

$$V = s^T M s. \tag{28}$$

Using the closed-loop eq. (11) and the property $\dot{M} - 2C$ skew-symmetric, we will have

$$\frac{d}{dt}V = -2s^T K s + 2s^T \epsilon. \tag{29}$$

Note that

$$\frac{d}{dt}V \leq -2\lambda_{\min}(K)\|s\|^2 + 2\|s\|\epsilon_m. \tag{30}$$

Using the comparison lemma [7], we will have

$$\|s\| \leq \sqrt{\frac{\lambda_{\max}(M)}{\lambda_{\min}(M)}} e^{-\frac{K}{\lambda_{\max}(M)}t}\|s(0)\| + \frac{\lambda_{\max}(M)}{\lambda_{\min}(K)\lambda_{\min}(M)}(1 - e^{-\frac{K}{\lambda_{\max}(M)}t}) \cdot \epsilon_m. \tag{31}$$

Therefore s will exponentially converge to

$$\frac{\lambda_{\max}(M)}{\lambda_{\min}(K)\lambda_{\min}(M)} \cdot \epsilon_m \tag{32}$$

Since $s = \dot{\tilde{q}} + \Lambda\tilde{q}$, $\|\tilde{q}\|$ will exponentially converge to

$$\frac{\lambda_{\max}(M)}{\lambda_{\min}(\Lambda)\lambda_{\min}(K)\lambda_{\min}(M)} \cdot \epsilon_m. \tag{33}$$

Moreover, since

$$\|\dot{\tilde{q}}\| \leq \|s\| + \lambda_{\max}(\Lambda)\|\tilde{q}\|, \tag{34}$$

$\|\dot{\tilde{q}}\|$ will converge to

$$\left(\frac{\lambda_{\max}(M)}{\lambda_{\min}(K)\lambda_{\min}(M)} + \frac{\lambda_{\max}(\Lambda)\lambda_{\max}(M)}{\lambda_{\min}(\Lambda)\lambda_{\min}(K)\lambda_{\min}(M)}\right) \cdot \epsilon_m. \tag{35}$$

Recall that $\tilde{x} = [\tilde{q}, \dot{\tilde{q}}]$. Thus finally we have the following upper bound of the error ball:

$$\|\tilde{x}\| \to \frac{\lambda_{\max}(M)}{\lambda_{\min}(K)\lambda_{\min}(M)} \sqrt{\left(\frac{1}{\lambda_{\min}(\Lambda)}\right)^2 + \left(1 + \frac{\lambda_{\max}(\Lambda)}{\lambda_{\min}(\Lambda)}\right)^2} \cdot \epsilon_m. \tag{36}$$

□

A.2.5 Proof of Theorem 3

Proof. We make the following assumption. $\mathcal{X}$ is the final set of target data we want to investigate. It can be the full state space in control. We define r_t as the regret of the predictor in timestep t. For robust regression predictor in each step, even though we model each step as a distribution shift and regard current training data as training and the selected next trajectory as testing, we assume we eventually apply the predictor f_t on the full target data $\mathcal{X}$ and can obtain the regret $r_t(x) = f_t(x) = f_t(x) - f^*(x)$ for $x \in \mathcal{X}$. Assuming r_0 is the regret of the base distribution, which is a constant.

$$\begin{aligned}
r_t &= r_t - r_{t-1} + r_{t-1} - r_{t-2} + ... + r_1 - r_0 + r_0 && (37)\\
&= (r_t - r_{t-1}) + (r_{t-1} - r_{t-2}) + ... + (r_1 - r_0) + r_0 && (38)\\
&= (f_t(x) - f_{t-1}(x)) + ... + (f_1(x) - f_0(x)) + r_0 && (39)\\
&\leq \sum_t \sqrt{2(\sigma_t(x) - \sigma_{t-1}(x)) \ln \frac{|\mathcal{X}|}{2\delta}} + \sum_t (\mu_t(x) - \mu_{t-1}(x)) + r_0 && (40)\\
&\leq \sum_t \sqrt{2(\sigma_t(x) - \sigma_{t-1}(x)) \ln \frac{|\mathcal{X}|}{2\delta}} + (\mu_t(x) - \mu_0(x)) + r_0 && (41)\\
&= \sum_t \sqrt{2(\sigma_t(x) - \sigma_{t-1}(x)) \ln \frac{|\mathcal{X}|}{2\delta}} + (\mu_t(x) - \mu^*(x), && (42)
\end{aligned}$$

with probability $1 - \delta$. This is due to the difference of two Gaussian distribution is also Gaussian. If R_t is the cumulative regret $\sum_t r_t$, then if we have have $R_t \leq t\epsilon$, then$\|f_t(x) - f^*(x)\| \leq \epsilon$. □

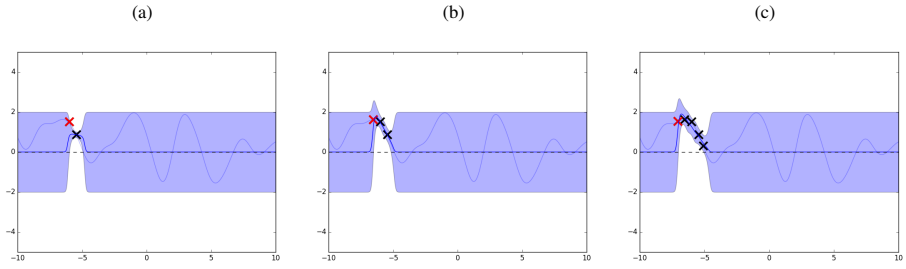

Figure 6: Experimental results of robust regression for stateless optimization on (a) the second, (b) the third, and (c) the fifth iteration.

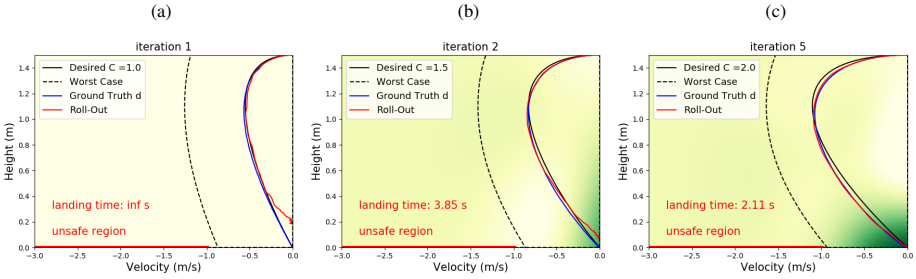

Figure 7: Experimental results on landing using Gaussian Processes with phase portrait of (a) the first iteration, (b) the second iteration, and (c) the fifth iteration.

A.2.6 Proof of Corollary 3

Proof. If there exist a desired trajectory $x_d^*(t)$ in the set, such that if we have dynamics model $\|f_t(x) - f^*(x)\| \leq \epsilon$ for all $x \in \mathcal{X}$, it is safe and would be selected and executed. Then if at timestep t, we have our dynamics model converge to the optimal in its set, but $x_d^*(t)$ is not selected, it would be selected in the next timestep, since Algorithm 2 select the most aggressive trajectory that is safe. Then according to Theorem 2, it would be tracked with $\|x'(t) - x_d^*(t)\| \leq \gamma\epsilon$. □

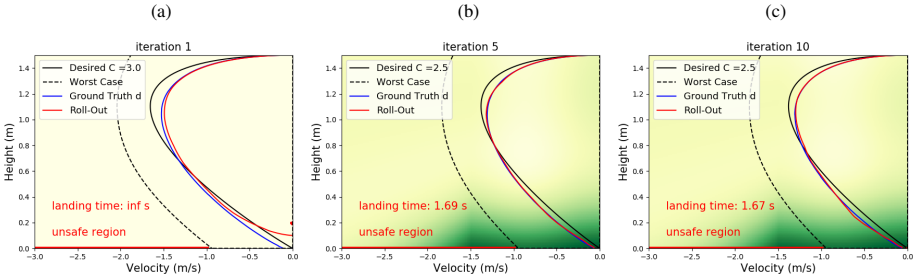

Figure 8: Experimental results on landing using Gaussian Processes with phase portrait of (a) the first iteration, (b) the fifth iteration, and (c) the tenth iteration.

A.3 Robust regression for stateless experimental design

Robust regression under covariate shifts can also be applied to standard stateless experimental design without dynamics models involved. We did some synthetic experiments to show its difference from GP based model such as [11]. We use a random generated mixture of Gaussian as unknown function we would like to optimize. Figure 6 shows a safe exploration process using “safe-UCB", which means it only explores the maximum point under the current prediction when its lower bound is safe.

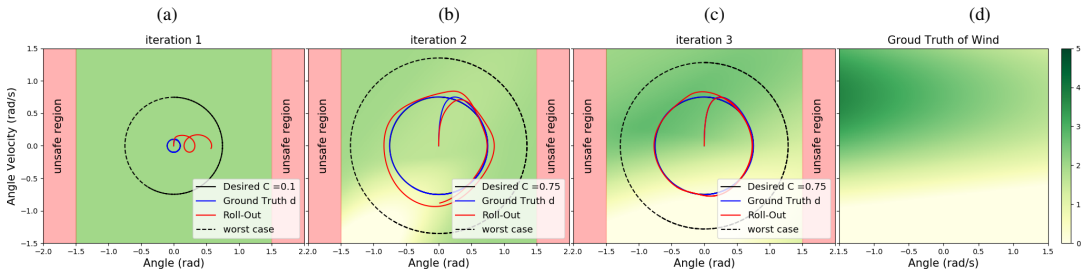

Figure 9: Experimental results on invert pendulum using Gaussian Processes with ground truth wind in (d).

A.4 Additional Experimental Results

We provide additional experimental result for Gaussian Processes. Figure 7 and Figure 8 are two sets of results on landing with GP with different kernel parameters.

We can see from the colormap that the prediction matches with the ground truth only in a small region but GP fails to generalize to boarder space of states. This is also a reason why it converges slower than the proposed method. Figure 9 shows GP's learning and exploration process under a different wind condition on invert pendulum.

A.5 Additional Experimental Details

We include our code for experiments in the supplemental materials, which include the simulation environment of the two control examples in this paper, the robust regression algorithm, and also GP implementation using GPytorch. We run our experiments on machine with processor 2.3 GHz Intel Core i5 and memory 8 GB 2133 MHz LPDDR3. The running time for each iteration is usually less than 2 seconds for GP-based version and less than 5 seconds for robust regression based version, without result visualization.