Article

The International Journal of Robotics Research
2021, Vol. 40(2-3) 574–593

DOI: 10.1177/0278364920959444
journals.sagepub.com/home/ijr
SAGE

PAC-Bayes control: learning policies that provably generalize to novel environments

Anirudha Majumdar[1], Alec Farid[1] and Anoopkumar Sonar[2]

Abstract

Our goal is to learn control policies for robots that provably generalize well to novel environments given a dataset of example environments. The key technical idea behind our approach is to leverage tools from generalization theory in machine learning by exploiting a precise analogy (which we present in the form of a reduction) between generalization of control policies to novel environments and generalization of hypotheses in the supervised learning setting. In particular, we utilize the probably approximately correct (PAC)-Bayes framework, which allows us to obtain upper bounds that hold with high probability on the expected cost of (stochastic) control policies across novel environments. We propose policy learning algorithms that explicitly seek to minimize this upper bound. The corresponding optimization problem can be solved using convex optimization (relative entropy programming in particular) in the setting where we are optimizing over a finite policy space. In the more general setting of continuously parameterized policies (e.g., neural network policies), we minimize this upper bound using stochastic gradient descent. We present simulated results of our approach applied to learning (1) reactive obstacle avoidance policies and (2) neural network-based grasping policies. We also present hardware results for the Parrot Swing drone navigating through different obstacle environments. Our examples demonstrate the potential of our approach to provide strong generalization guarantees for robotic systems with continuous state and action spaces, complicated (e.g., nonlinear) dynamics, rich sensory inputs (e.g., depth images), and neural network-based policies.

Keywords

Learning-based control, generalization, safety

1. Introduction

Imagine an unmanned aerial vehicle (UAV) that successfully navigates a thousand different obstacle environments or a robotic manipulator that successfully grasps a million objects in our dataset. How likely are these systems to succeed on a novel (i.e., previously unseen) environment or object? How can we explicitly learn control policies that provably generalize well to environments or objects that our robot has not encountered previously? Current approaches for designing control policies for robotic systems either do not provide such guarantees on generalization or provide guarantees only under very restrictive assumptions (e.g., strong assumptions on the geometry of a novel environment (Althoff et al., 2015; Fraichard, 2007; Majumdar and Tedrake, 2017; Schouwenaars et al., 2004)).

The goal of this article is to develop an approach for learning control policies for robotic systems that provably generalize well with high probability to novel environments given a dataset of example environments. The key conceptual idea for enabling this is to establish a precise analogy between generalization of policies to novel environments and generalization in supervised learning. This analogy allows us to translate techniques for learning hypotheses with generalization guarantees in the supervised learning setting into techniques for learning control policies for robot tasks with performance guarantees on novel environments.

In order to obtain more insight into this analogy, suppose we have a dataset of N objects. A simple approach to learning a grasping policy is to synthesize one that achieves the best possible performance on these N objects. However,

[1]Department of Mechanical and Aerospace Engineering,Princeton University, Princeton, NJ, USA
[2]Department of Computer Science, Princeton University,Princeton, NJ, USA

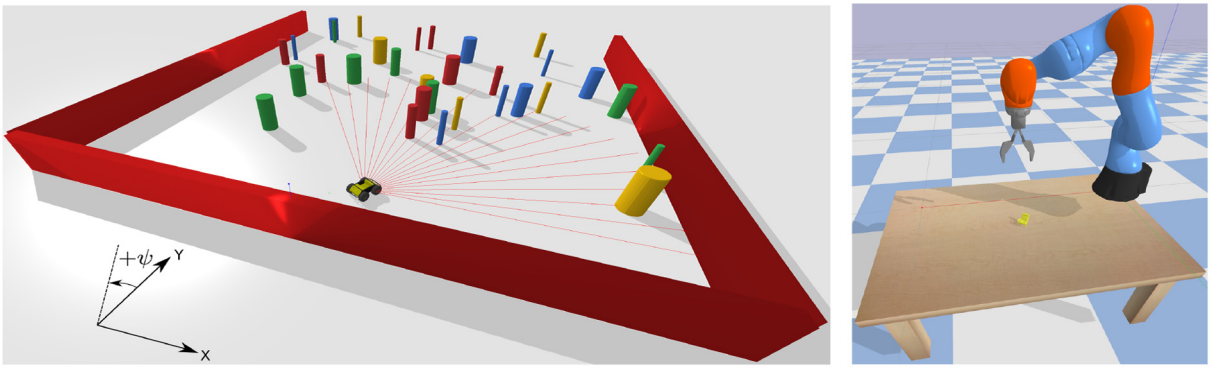
Corresponding author:
Anirudha Majumdar, Department of Mechanical and Aerospace Engineering,Princeton University, D202-B EQuad, Olden Street, Princeton, NJ 08544, USA.
Email: ani.majumdar@princeton.edu

such a strategy might result in an overly complex policy that overfits to the specific objects at hand. This is a particularly important challenge for robotics applications since datasets are generally relatively small (e.g., as compared with training sets for image classification tasks). In order to learn a policy that generalizes well to novel environments, we may need to add a "regularizer" that penalizes the "complexity" of the policy. This raises the following questions. (1) What form should this regularizer take? (2) Can we provide a formal guarantee on the performance of the resulting policy on novel environments?

The analogous questions for *supervised* learning algorithms have been studied extensively in the literature on *generalization theory* in machine learning. Here we leverage probably approximately correct (PAC)-Bayes theory (McAllester, 1999), which provides some of the tightest known generalization bounds for classical supervised learning approaches (Germain et al., 2009; Langford and Shawe-Taylor, 2003; Seeger, 2002). Very recently, PAC-Bayes analysis has also been used to train deep neural networks with guarantees on generalization performance (Dziugaite and Roy, 2017a; Neyshabur et al., 2017a,b). As we show, we can leverage PAC-Bayes theory to provide precise answers to both questions posed previously; it will allow us to specify a regularizer for designing (stochastic) control policies that provably generalize well (with high probability) to novel environments.

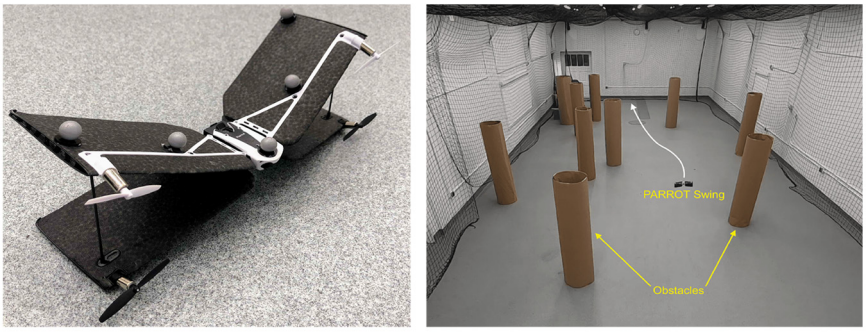
1.1. Statement of contributions

The primary contribution of this article is the introduction of a framework for providing *generalization guarantees* for learning-based control of robots. While generalization bounds have been studied extensively in the literature on supervised learning (as discussed previously), there has been relatively little work on this topic in the literature on robot learning (see Section 1.2 for a thorough literature review). To the best of the authors' knowledge, the results in this article constitute the first attempt to provide generalization guarantees on learning-based control policies for robotic systems with continuous state and action spaces, complicated (e.g., nonlinear or hybrid) dynamics, and rich sensory inputs (e.g., RGB-D images). To this end, this article makes four specific contributions. First, we provide a *reduction* that allows us to translate generalization bounds for supervised learning problems to generalization bounds for control policies. We apply this reduction to translate PAC-Bayes bounds to the control setting we consider here (Section 4). Second, we propose learning algorithms that minimize the regularized cost functions specified by PAC-Bayes theory in order to synthesize control policies with generalization guarantees (Section 5). In the setting where we are optimizing over a finite policy space (Section 5.1), the corresponding optimization problem can be solved using *convex* optimization techniques (*relative entropy programs* (REPs) in particular). In the more general setting of continuously parameterized policies (Section 5.2), we rely on stochastic gradient descent (SGD) to perform the optimization. Third, in Section 6.2 we present an extension of our basic approach that allows us to learn policies that are distributionally robust (i.e., handle settings where test environments are drawn from a different distribution than training environments). Fourth, we demonstrate our approach in simulation for learning (i) depth sensor-based reactive obstacle avoidance policies for the ground robot model shown in Figure 1(a) (Section 7.1), and (ii) neural network-based grasping policies for the manipulator model shown in Figure 1(b) (Section 7.2). Finally, we also present hardware results for reactive obstacle avoidance control with the Parrot Swing drone shown in Figure 2(a) (Section 8). Our



(a) (b)

Figure 1. We demonstrate our approach for learning (i) reactive obstacle avoidance policies for a differential drive ground vehicle model equipped with a depth sensor, and (ii) neural network-based grasping policies for a manipulator model equipped with an RGB-D sensor. Our approach provides strong guarantees on the performance of the learned policies on novel environments even with a relatively small number of training environments (e.g., a guaranteed expected collision-free traversal rate of 87.9% using 1,000 training environments for the obstacle avoidance example and a guaranteed expected success rate of 70.6% for the grasping example using 2,000 training objects).



(a) (b)

Figure 2. (a) Parrot Swing drone: a quadrotor/fixed-wing hybrid vehicle. We demonstrate our approach for learning reactive obstacle avoidance policies for the Swing given a simulated depth sensor. Our approach provides a guaranteed expected collision-free traversal rate of 88.6% on novel environments using 1,000 simulated training environments. When testing on unseen environments within the netted area pictured in (b), the Swing succeeds in 18/20 trials. Videos of representative trials can be found at https://youtu.be/p5CjcSsojg8.

simulation and hardware results demonstrate that we are able to obtain strong generalization guarantees even with a relatively small number of training environments. We compare the bounds obtained from PAC-Bayes theory with exhaustive sampling to illustrate the tightness of the bounds.

A preliminary version of this work (Majumdar and Goldstein, 2018) was presented at the Conference on Robot Learning (CoRL) 2018. In this significantly revised and extended version, we additionally present: (i) an extension of our basic approach for providing generalization guarantees in settings where test environments are drawn from a different distribution to training environments (Section 6.2); (ii) an application of our framework for learning neural network-based grasping policies (Section 7.2); (iii) a method for handling stochastic dynamics (Section 6.1); (iv) hardware implementation of the depth sensor-based reactive obstacle avoidance policies (Section 8); and (v) a more thorough discussion of the challenges associated with our approach and promising future directions (Section 9).

1.2. Related work

One approach for synthesizing control policies with guaranteed performance is to leverage robust control techniques (e.g., H^∞ control (Francis, 1987) or chance-constrained programming (Blackmore et al., 2006; Charnes and Cooper, 1959; Ono et al., 2015; Vitus and Tomlin, 2011)). However, such techniques typically require an explicit description of the uncertainty affecting the system. While uncertainty models for the robot's dynamics or measurements can often be obtained via system identification, assuming an uncertainty model for the environment (e.g., a distribution over all possible environment geometries) is unrealistic. One way to address this is to assume that a novel environment satisfies conditions that allow a real-time planner to *always* succeed. For example, in the context of navigation, this constraint could be satisfied by hand-coding emergency maneuvers (e.g., stopping maneuvers or loiter circles) that are always guaranteed to succeed (Althoff et al., 2015; Fraichard, 2007; Schouwenaars et al., 2004). However, requiring the existence of such emergency maneuvers can lead to extremely conservative behavior. Another approach is to assume that the environment satisfies certain geometric conditions (e.g., large separation between obstacles) that allow for safe navigation (Majumdar and Tedrake, 2017). However, such conditions are rarely satisfied by real-world environments. Moreover, such conditions are domain specific; it is not clear how one would specify such constraints for problems other than navigation (e.g., grasping).

Another conceptually appealing approach for synthesizing policies with guaranteed performance on a priori unknown environments is to model the problem as a partially observable Markov decision process (POMDP) (Kaelbling et al., 1998), where the environment is part of the (partially observed) state of the system (Richter et al., 2015). Computational considerations aside, such an approach is made infeasible by the need to specify a distribution over environments the robot might encounter. Unfortunately, specifying such a distribution over real-world environments is an extremely challenging endeavor. Thus, many approaches (including ours) assume that we only have *indirect* access to the true underlying distribution over environments in the form of examples. For example, Richter and Roy (2017); Richter et al. (2015) proposed an approximation to the POMDP framework in the context of navigation by learning to predict future collision probabilities from past data. The work on deep-learning-based approaches for control represents another prominent set of techniques where interactions with example environments are used to learn control policies (see, e.g., Agrawal et al., 2016; Gupta et al., 2017a,b; Lenz et al., 2015; Levine et al., 2016; Mahler et al., 2017; Sünderhauf et al., 2018;

Tobin et al., 2017; Zhu et al., 2017). While the approaches mentioned previously have led to impressive empirical demonstrations, it is challenging to guarantee that such methods will perform well on environments that are not part of the training data (especially when a limited number of training examples are available, as is often the case for robotics applications). Our work seeks to address this challenge using ideas from generalization theory.

The primary theoretical framework we utilize in this paper is PAC-Bayes generalization theory (McAllester, 1999). PAC-Bayes theory provides some of the tightest known generalization bounds for classical supervised learning problems (Germain et al., 2009; Langford and Shawe-Taylor, 2003; Seeger, 2002) and has recently been applied to explain and promote generalization in deep learning (Dziugaite and Roy, 2017a; Neyshabur et al., 2017a,b). PAC-Bayes theory has also been applied to learn control policies for Markov decision processes (MDPs) with provable sample complexity bounds (Fard and Pineau, 2010; Fard et al., 2012). These approaches also exploit the intuition (see Section 1) that "regularizing" policies in an appropriate manner can prevent overfitting and lead to sample efficiency (see also Bagnell, 2004; Bagnell and Schneider, 2001; Kearns et al., 2000; Neu et al., 2017; Schulman et al., 2015 for other approaches that exploit this intuition in the reinforcement learning context). However, we note that the focus of our work is quite different from the work on PAC-Bayes MDP bounds (and the more general framework of PAC MDP bounds (Brafman and Tennenholtz, 2002; Fu and Topcu, 2014; Kearns and Singh, 2002)), which consider the standard reinforcement learning setup where a control policy must be learned through multiple interactions with a given MDP (with unknown transition dynamics and/or rewards). In contrast, here we focus on *zero-shot* generalization to a novel environment (e.g., obstacle environments or objects). In other words, a policy learned from examples of different environments must immediately perform well on a new one (i.e., without further exploratory interactions with the new environment). We further note that Fard and Pineau (2010) considered finite state and action spaces along with policies that depend on full state feedback whereas (Fard et al., 2012) relaxed the assumption on finite state spaces but retained the other modeling assumptions. In contrast, we target systems with continuous state and action spaces and synthesize control policies that rely on rich sensory inputs.

On the algorithmic front, we make significant use of REPs (Chandrasekaran and Shah, 2017). REPs constitute a rich class of *convex* optimization problems that generalize many other problems including linear programs, geometric programs, and second-order cone programs (Boyd and Vandenberghe, 2004). REPs are optimization problems in which a linear functional of the decision variables is minimized subject to linear constraints and conic constraints given by a *relative entropy cone*. REPs are amenable to efficient solution techniques (e.g., interior point methods (Nesterov and Nemirovskii, 1994)) and can be solved using existing software packages (e.g., Mosek (MOSEK ApS, 2019), SCS (O'Donoghue et al., 2016, 2017), and ECOS (Domahidi et al., 2013)). We refer the reader to Chandrasekaran and Shah (2017) for a more thorough introduction to REPs. Importantly for us, REPs can handle constraints of the form $\mathbb{D}(p \parallel q) \leqslant c$, where p and q are decision variables corresponding to probability vectors, $\mathbb{D}(\cdot \parallel \cdot)$ represents the Kullback–Leibler (KL) divergence, and c is a scalar decision variable. As we show, this allows us to use REPs to learn control policies using the PAC-Bayes framework in the setting where we are optimizing over a finite set of policies.

1.3. Notation

We use the notation $v[i]$ to refer to the ith component of a vector $v \in \mathbb{R}^n$. We use $\mathbb{R}^n_+$ to denote the set of elementwise non-negative vectors in $\mathbb{R}^n$, $\mathbb{Z}_+$ to denote non-negative integers, and $\odot$ to denote element-wise multiplication.

2. Problem formulation

We assume that the robot's dynamics are described by a discrete-time system:

$$x(t+1) = f(x(t), u(t); E) \tag{1}$$

where $t \in \mathbb{Z}_+$ is the time index, $x(t) \in \mathcal{X}$ is the state at time t, $u(t) \in \mathcal{U}$ is the control input at time t, and E is the environment that the robot operates in. We use the term "environment" here broadly to refer to any factors that are external to the robot. For example, E could refer to an obstacle field that a mobile robot is attempting to navigate through, external disturbances (e.g., wind gusts) that a UAV is subjected to, or an object that a manipulator is attempting to grasp.

Let $\mathcal{E}$ denote the space of all possible environments. We then make the following assumption.

Assumption 1. *There is an underlying distribution $\mathcal{D}$ over $\mathcal{E}$ from which environments are drawn.*

Importantly, we *do not* assume that we have explicit descriptions of $\mathcal{E}$ or $\mathcal{D}$. Instead, we only assume indirect access to $\mathcal{D}$ in the form of a dataset $S = \{E_1, \ldots, E_N\}$ of N training environments drawn independent and identically distributed from $\mathcal{D}$. In Section 6.2, we present an extension of our basic framework that allows us to relax this assumption and handle settings where training and test environments are drawn from different distributions.

Let $g : \mathcal{X} \times \mathcal{E} \to \mathcal{Y}$ denote the robot's sensor mapping from a state x and an environment E to an observation $y = g(x; E) \in \mathcal{Y}$. Since we are interested in partially observable settings, we do not make any particular assumptions (e.g., injectivity or bijectivity) on the sensor mapping g. Let $\pi : \mathcal{Y} \to \mathcal{U}$ denote a control policy that maps sensor measurements to control inputs. Note that this is a very general model and can capture control policies that depend on *histories* of sensor measurements (by simply

augmenting the state to keep track of histories of states and letting $\mathcal{Y}$ denote the space of histories of sensor measurements).

We assume that the robot's desired behavior is encoded through a cost function. In particular, let $r_\pi : \mathcal{E} \to (\mathcal{X} \times \mathcal{U})^T$ denote the function that "rolls out" the system with control policy π, i.e., r_π maps an environment E to the state-control trajectory one obtains by applying the control policy π (up to a time horizon T). We will assume that the environment captures all sources of stochasticity (including random initial conditions) and the rollout function for a *particular* environment is thus deterministic (we discuss the case of stochastic rollouts in Section 6.1). We then let $C(r_\pi; E)$ denote the cost incurred by control policy π when operating in environment E over a time horizon T. We assume that the cost $C(r_\pi; E)$ is bounded and will assume (without further loss of generality) that $C(r_\pi; E) \in [0, 1]$. We make the following important assumption in this work.

Assumption 2. *Given any control policy π, we can compute the cost $C(r_\pi; E_i)$ for the training environments $E_1, \ldots, E_N$.*

This assumption is satisfied if one can simulate the robot's operation in the environments $E_1, \ldots, E_N$. We note that computational considerations aside, we do not make any restrictions on the dynamics f or the sensor mapping g beyond the ability to simulate them. The models that our approach can handle are, thus, extremely rich in principle (e.g., nonlinear or hybrid dynamics, sensor models involving raycasting or simulated vision, etc.).

Another possibility for satisfying Assumption 2 is to run the policy π on the hardware system itself in the given environments. This may be a feasible option for problems such as grasping, which are not safety-critical in nature. In such cases, our approach does not require models of the dynamics, sensor mapping, or the rollout function.

Goal: Our goal is to design a control policy that minimizes the expected value of the cost C across environments:

$$\min_{\pi \in \Pi} \; C_{\mathcal{D}}(\pi) := \min_{\pi \in \Pi} \; \mathop{\mathbb{E}}_{E \sim \mathcal{D}} [C(r_\pi; E)] \tag{2}$$

In this work, it will be useful to consider a more general setting where we choose a *distribution* P over the control policy space Π instead of making a single deterministic choice. This is because the PAC-Bayes bounds we use will assume this setting. Our goal is then to solve the following optimization problem, which we refer to as $\mathcal{OPT}$:

$$C^\star := \min_{P \in \mathcal{P}} \; C_{\mathcal{D}}(P) := \min_{P \in \mathcal{P}} \; \mathop{\mathbb{E}}_{E \sim \mathcal{D}} \; \mathop{\mathbb{E}}_{\pi \sim P} [C(r_\pi; E)] \; (\mathcal{OPT})$$

where $\mathcal{P}$ denotes the space of probability distributions over Π. Note that the outer expectation here is taken with respect to the *unknown* distribution $\mathcal{D}$. This constitutes the primary challenge in tackling this problem.

3. Background

The primary technical framework we leverage in this article is PAC-Bayes theory. In Section 3.2, we provide a brief overview of the key results from PAC-Bayes theory in the context of supervised learning. We first provide some brief background on the properties of the KL divergence in Section 3.1 and show how we can compute its inverse using relative entropy programming in Section 3.1.1.

3.1. KL divergence

Given two discrete probability distributions P and Q defined over a common set, the KL divergence from Q to P is defined as

$$\mathbb{D}(P \parallel Q) := \sum_i P[i] \log\left(\frac{P[i]}{Q[i]}\right) \tag{3}$$

For scalars $p, q \in [0, 1]$, we define

$$\mathbb{D}(p \parallel q) := \mathbb{D}(B(p) \parallel B(q)) = p \log\frac{p}{q} + (1-p)\log\frac{1-p}{1-q} \tag{4}$$

where $B(p)$ denotes a Bernoulli distribution on $\{0, 1\}$ with parameter (i.e., mean) p.

For distributions P and Q of a continuous random variable, the KL divergence is defined to be

$$\mathbb{D}(P \parallel Q) = \int p(x) \log\frac{p(x)}{q(x)}\, dx \tag{5}$$

where p and q denote the densities of P and Q. Importantly, if P and Q correspond to normal distributions $N_p = \mathcal{N}(\mu_p, \Sigma_p)$ and $N_q = \mathcal{N}(\mu_q, \Sigma_q)$ over $\mathbb{R}^d$, the KL divergence can be computed in closed form as

$$\mathbb{D}(N_p \parallel N_q) = \frac{1}{2} \left(\mathrm{Tr}(\Sigma_q^{-1}\Sigma_p) + (\mu_q - \mu_p)^T \Sigma_q^{-1}(\mu_q - \mu_p) + \log\frac{\det(\Sigma_q)}{\det(\Sigma_p)} - d\right) \tag{6}$$

3.1.1. Computing KL inverse using relative entropy programming. PAC-Bayes bounds (Section 3.2) are typically expressed as bounds on a quantity $q^\star \in [0, 1]$ of the form $\mathbb{D}(p \parallel q^\star) \leqslant c$ (for some $p \in [0, 1]$ and $c \geqslant 0$). These bounds can then be used to upper bound $q^\star$ by the *KL inverse* as follows:

$$q^\star \leqslant \mathbb{D}^{-1}(p \parallel c) := \sup\{q \in [0, 1] | \mathbb{D}(p \parallel q) \leqslant c\} \tag{7}$$

In prior work on PAC-Bayes theory, the KL inverse was numerically approximated using local root-finding

techniques such as Newton's method (Dziugaite and Roy, 2017a,b), which do not have a priori guarantees on convergence to a global solution. Here we observe that the KL inverse is readily expressed as the optimal value of a simple REP (see Section 1.2). In particular, the expression for the KL inverse in (7) corresponds to an optimization problem with a (scalar) decision variable q, a linear cost function (i.e., $-q$), linear inequality constraints (i.e., $0 \leqslant q \leqslant 1$), and a constraint on the KL divergence between the decision variable q and the constant p. We can thus compute the KL inverse exactly (up to numerical tolerances) using convex optimization (e.g., interior point methods (Chandrasekaran and Shah, 2017)).

3.2. PAC-Bayes theory in supervised learning

We now provide a brief overview of the key results from PAC-Bayes theory in the context of supervised learning. Let $\mathcal{Z}$ be an input space and $\mathcal{Z}'$ be a set of labels. Let $\mathcal{D}$ be the (unknown) true distribution on $\mathcal{Z}$. Let $\mathcal{H}$ be a hypothesis class consisting of functions $h_w : \mathcal{Z} \to \mathcal{Z}'$ parameterized by $w \in \mathbb{R}^d$ (e.g., neural networks parameterized by weights w). Let $l : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}$ be a loss function.[1] We denote by $\mathcal{P}$ the space of probability distributions on the parameter space $\mathbb{R}^d$. Informally, we refer to distributions on $\mathcal{H}$ when we mean distributions over the underlying parameter space.

PAC-Bayes analysis then applies to learning algorithms that output a *distribution* over hypotheses. Specifically, the PAC-Bayes framework applies to learning algorithms with the following structure.

1. Choose a *"prior" distribution* $P_0 \in \mathcal{P}$ *before* observing any data.
2. Observe training data samples $S = \{z_i\}_{i=1}^N$ and choose a *posterior distribution* $P \in \mathcal{P}$. This posterior can depend on the data and the prior.

It is important to note that the posterior distribution P *need not* be the Bayesian posterior. PAC-Bayes theory applies to *any* distribution P.

Let us denote the training loss associated with the posterior distribution P as

$$l_S(P) := \frac{1}{N} \sum_{z \in S} \mathop{\mathbb{E}}_{w \sim P}[l(h_w; z)] \tag{8}$$

and the true expected loss as

$$l_{\mathcal{D}}(P) := \mathop{\mathbb{E}}_{z \sim \mathcal{D}} \mathop{\mathbb{E}}_{w \sim P}[l(h_w; z)] \tag{9}$$

The following theorem is the primary result from PAC-Bayes theory.[2]

Theorem 1 (PAC-Bayes bound for supervised learning (Maurer, 2004; McAllester, 1999)). *For any* $\delta \in (0, 1)$*, with probability at least* $1 - \delta$ *over samples* $S \sim \mathcal{D}^N$*, the following inequality holds*:

$$\mathbb{D}(l_S(P) \parallel l_{\mathcal{D}}(P)) \leqslant \frac{\mathbb{D}(P \parallel P_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{N} \tag{10}$$

Here, $\mathbb{D}(l_S(P) \parallel l_{\mathcal{D}}(P))$ is interpreted as a KL divergence between Bernoulli distributions and computed using (4) (this is meaningful since $l_S(P)$ and $l_{\mathcal{D}}(P)$ are scalars bounded within $[0, 1]$).

Intuitively, Theorem 1 provides a bound on how "close" the training loss $l_S(P)$ and the true expected loss $l_{\mathcal{D}}(P)$ are. However, in practice, one would like to find an *upper bound* on the true expected loss $l_{\mathcal{D}}(P)$. Such an upper bound can be obtained by computing the KL inverse (see Section 3.1.1):

$$l_{\mathcal{D}}(P) \leqslant \mathbb{D}^{-1}\left(l_S(P) \parallel \frac{\mathbb{D}(P \parallel P_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{N}\right) \tag{11}$$

Another upper bound that is useful for the purpose of optimization is provided by the following corollary, which follows from Theorem 1 by applying the well-known upper bound for the KL inverse, by applying Pinsker's inequality one obtains $\mathbb{D}^{-1}(p \parallel c) \leqslant p + \sqrt{c/2}$.

Corollary 1 (PAC-Bayes upper bound for supervised learning (Maurer, 2004; McAllester, 1999)). *For any* $\delta \in (0, 1)$*, with probability at least* $1 - \delta$ *over samples* $S \sim \mathcal{D}^N$*, the following inequality holds*:

$$\underbrace{l_{\mathcal{D}}(P)}_{\text{True expected loss}} \leqslant \underbrace{l_S(P)}_{\text{Training loss}} + \underbrace{\sqrt{\frac{\mathbb{D}(P \parallel P_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}}}_{\text{``Regularizer''}} \tag{12}$$

Corollary 1 provides a strategy for choosing a distribution P over hypotheses with a provable guarantee on generalization: minimize the right-hand side (RHS) of inequality (12) consisting of the training loss and a "regularization" term.

4. PAC-Bayes control

We now describe our approach for adapting the PAC-Bayes framework in order to tackle the policy learning problem $\mathcal{OPT}$ and synthesize (stochastic) control policies with guaranteed expected performance across novel environments. Our key idea for doing this is to exploit a precise analogy between the supervised learning setting from Section 3.2 and the policy learning setting described in Section 2. Table 1 presents this relationship.

One can think of the relationship in Table 1 as providing a *reduction* from the policy learning problem $\mathcal{OPT}$ to a supervised learning problem. We are provided input data in

Table 1. A reduction from the control policy learning problem we consider here to the supervised learning setting

Supervised learning			Policy learning	
Input data	$z \in \mathcal{Z}$	⟵	Environment	$E \in \mathcal{E}$
Hypothesis	$h_w : \mathcal{Z} \to \mathcal{Z}'$		Rollout function	$r_\pi : \mathcal{E} \to (\mathcal{X} \times \mathcal{U})^H$
Loss	$l(h_w; z)$		Cost	$C(r_\pi; E)$

the form of a data set of example environments. Choosing a "hypothesis" corresponds to choosing a control policy π (since the rollout function r_π is determined by π). A "hypothesis" maps an environment E to a "label," corresponding to the state-control trajectory obtained by applying π on E. This "label" incurs a loss $C(r_\pi; E)$.

We can use this reduction to translate the PAC-Bayes theorems for supervised learning (Theorem 1 and Corollary 1) to the control setting. Similar to the supervised learning setting, we assume that the space Π of control policies is parameterized by $w \in \mathbb{R}^d$. This, in turn, produces a parameterization of rollout functions. With a slight abuse of notation, we refer to rollout functions r_w instead of r_π (with the understanding that w is the parameter vector for the control policy π).

Let P_0 be a "prior" distribution over the parameter space $\mathbb{R}^d$ chosen before seeing any example environments. The prior can be used to encode domain knowledge, but need not be "true" in any Bayesian sense (i.e., bounds will hold for any prior). Let P be a (possibly data-dependent) "posterior." Following the notation from Section 2, we denote the true expected cost across environments by $C_{\mathcal{D}}(P)$. We denote the cost on the training environments as

$$C_S(P) := \frac{1}{N} \sum_{E \in S} \mathop{\mathbb{E}}_{w \sim P}[C(r_w; E)] \tag{13}$$

The following theorem is then an exact analogy of Corollary 1.

Theorem 2 (PAC-Bayes bound for control policies). *For any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over sampled environments $S \sim \mathcal{D}^N$, the following inequality holds*:

$$\underbrace{C_{\mathcal{D}}(P)}_{\text{True expected cost}} \leqslant C_{PAC}(P) := \underbrace{C_S(P)}_{\text{Training cost}} + \underbrace{\sqrt{\frac{\mathbb{D}(P \parallel P_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}}}_{\text{``Regularizer''}} \tag{14}$$

Proof. The proof follows immediately from Corollary 1 given the reduction in Table 1. □

This theorem constitutes our primary tool for learning policies with guarantees on their expected performance across novel environments. In particular, the left-hand side of inequality (14) is the cost function $C_{\mathcal{D}}(P)$ of the optimization problem $\mathcal{OPT}$. Theorem 2 thus provides an upper bound (that holds with probability $1 - \delta$) on the true expected performance across environments of any policy distribution P in terms of the loss on the sampled environments in $S = \{E_i\}_{i=1}^N$ and a "regularizer." Our approach for choosing P is to minimize this upper bound. Algorithm 1 outlines the steps involved in our approach.

Algorithm 1 PAC-Bayes Policy Learning

1: Fix prior distribution $P_0 \in \mathcal{P}$ over policies
2: **Inputs:** $S = \{E_1, \ldots, E_N\}$: Training environments, δ: Probability threshold
3: **Outputs:**
4: $P^\star_{\text{PAC}} = \operatorname{argmin}_{P \in P} \; C_{PAC}(P) := \frac{1}{N} \sum_{E \in S} \mathop{\mathbb{E}}_{w \sim P}[C(r_w; E)] + \sqrt{\frac{\mathbb{D}(P \| P_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}}$
5: $C^\star_{bound} := \mathbb{D}^{-1}\left(C_S(P^\star_{\text{PAC}}) \parallel \frac{\mathbb{D}(P^\star_{\text{PAC}} \| P_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{N}\right)$

We note that while P is chosen by optimizing $C_{\text{PAC}}(P)$ (i.e., the RHS of inequality (14)), the final upper bound $C^\star_{\text{bound}}$ on $C_{\mathcal{D}}(P)$ is not computed as $C_{\text{PAC}}(P^\star_{\text{PAC}})$. While this is a valid upper bound, a tighter bound is provided by inequality (11). The observations made in Section 3.1.1 allow us to compute this final bound using a REP. This is the bound we report in the results presented in Section 7.

5. Computing PAC-Bayes control policies

We now describe how to tackle the optimization problem in Algorithm 1 for minimizing the upper bound on the true expected cost. We first discuss the setting where the control policy space Π is finite (Section 5.1). For this setting, the optimization problem can be solved to global optimality via relative entropy programming. We then tackle the more general setting where Π is continuously parameterized in Section 5.2.

5.1. Finite control policy space

Let the space of policies be $\Pi = \{\pi_1, \ldots, \pi_L\}$. Our goal is then to optimize a *discrete* probability distribution P (with corresponding probability vector p) over the space Π. Thus, $p[j]$ denotes the probability assigned to policy π_j. Define a matrix $\hat{C}$ of costs, where each element

$$\hat{C}[i, j] = C(r_{\pi_j}; E_i) \tag{15}$$

corresponds to the cost incurred on environment $E_i \in S$ by policy $\pi_j \in \Pi$ (recall that Assumption 2 implies that we can compute each $\hat{C}[i,j]$). The training cost from inequality (14) can then be written as

$$\frac{1}{N}\sum_{E\in S}\mathop{\mathbb{E}}_{\pi\sim P}[C(r_\pi;E)] = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L}\hat{C}[i,j]p[j] := \bar{C}p \quad (16)$$

where the matrix $\bar{C}$ is defined as

$$\bar{C} := \frac{1}{N}\mathbf{1}^T\hat{C} \quad (17)$$

Here, 1 is the all-ones vector of size $N \times 1$. We note that finding a vector p that minimizes the training cost corresponds to solving a *linear program*.

Minimizing the PAC-Bayes upper bound $C_{\mathrm{PAC}}(P)$ corresponds to solving the following optimization problem:

$$\begin{aligned}\min_{p\in\mathbb{R}^L} \quad & \bar{C}p + \sqrt{\frac{\mathbb{D}(p \parallel p_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}} \\ \text{s.t.} \quad & 0 \leqslant p \leqslant 1,\ \sum_j p[j] = 1\end{aligned} \quad (18)$$

This optimization problem can be *equivalently* reformulated via an *epigraph constraint* (Boyd and Vandenberghe, 2004) as

$$\begin{aligned}\min_{p\in\mathbb{R}^L,\tau} \quad & \tau \\ \text{s.t.} \quad & \tau \geqslant \bar{C}p + \sqrt{\frac{\mathbb{D}(p \parallel p_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}} \\ & 0 \leqslant p \leqslant 1,\ \sum_j p[j] = 1\end{aligned}$$

We further rewrite the problem as

$$\begin{aligned}\min_{p\in\mathbb{R}^L,\tau,\lambda} \quad & \tau \\ \text{s.t.} \quad & \lambda^2 \geqslant \frac{\mathbb{D}(p \parallel p_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N} \\ & \lambda = \tau - \bar{C}p,\ \lambda \geqslant 0 \\ & 0 \leqslant p \leqslant 1,\ \sum_j p[j] = 1\end{aligned} \quad (19)$$

Our key observation here is that for a *fixed* $\lambda = \lambda_0$, the previous problem is an REP since it consists of minimizing a linear cost function subject to linear equality and inequality constraints and an additional inequality constraint of the form $\mathbb{D}(p \parallel p_0) \leqslant \text{constant}$.

We note that $\lambda \in [0,1]$ since $\lambda = \tau - \bar{C}p$, where $\tau \in [0,1]$ (because τ upper bounds the true expected cost) and $\bar{C}p \in [0,1]$ (recall that we assumed that costs are bounded between 0 and 1). In order to solve problem (19) to global optimality, we can thus simply search over the one-dimensional parameter $\lambda \in [0,1]$ (e.g., by simply discretizing the interval $[0,1]$, performing a bisection search, etc.) and find the setting of λ that leads to the lowest optimal value for the corresponding REP.

5.2. Continuously parameterized control policy space

We now consider policies π_w parameterized by the vector $w \in \mathbb{R}^d$ (e.g., neural networks parameterized by weights). We consider stochastic policies defined by probability distributions over the parameters w. Here, we choose Gaussian distributions $w \sim \mathcal{N}(\mu, \Sigma)$ with diagonal covariance $\Sigma = \mathrm{diag}(s)$ (with $s \in \mathbb{R}^d_+$) and use the shorthand $\mathcal{N}_{\mu,s} := \mathcal{N}(\mu, \mathrm{diag}(s))$. Using Gaussians makes computations easier since we can express the KL divergence between Gaussians in closed form (see Section 3.1). We can then apply Algorithm 1 and choose μ, s to minimize the PAC-Bayes upper bound $C_{\mathrm{PAC}}(\mathcal{N}_{\mu,s})$. In order to turn this into a practical algorithm, there are two primary issues we need to address.

First, in order to minimize the bound $C_{\mathrm{PAC}}(\mathcal{N}_{\mu,s})$, one would like to apply gradient-based methods (e.g., SGD). However, the cost function may not be a differentiable function of the parameters w. For example, in the case of designing obstacle avoidance policies, a natural (but non-differentiable) cost function is one that assigns a cost of 1 if the robot collides (and 0 otherwise). To tackle this issue, we employ a differentiable surrogate for the cost function during optimization (note that the final bound is still evaluated for the original cost function). This surrogate will necessarily depend on the application at hand; we present examples in the contexts of obstacle avoidance and grasping in Section 7.

The second challenge is the fact that computing the training cost $C_S(\mathcal{N}_{\mu,s})$ requires computing the following expectation over policies:

$$\mathop{\mathbb{E}}_{w\sim\mathcal{N}_{\mu,s}}[C(r_w;E)]. \quad (20)$$

For most realistic settings, this expectation cannot be computed in closed form. We address this issue in a manner similar to Dziugaite and Roy (2017a). In particular, in order to optimize μ and s using gradient descent, we take gradient steps with respect to the following unbiased estimator of $C_S(\mathcal{N}_{\mu,s})$:

$$\frac{1}{N}\sum_{E\in S} C(r_{\mu+\sqrt{s}\odot\xi};E), \quad \xi\sim\mathcal{N}_{0,I_d} \quad (21)$$

In other words, in each gradient step we use an independent and identically distributed sample of ξ and compute the gradient of (21) with respect to μ and s.

At the end of the optimization procedure, we fix the optimal $\mu^\star$ and $s^\star$ and estimate the training cost $C_S(P) = C_S(\mathcal{N}_{\mu^\star,s^\star})$ by producing a large number of samples $w_1, \ldots, w_L$ drawn from $\mathcal{N}_{\mu^\star,s^\star}$:

Algorithm 2 PAC-Bayes Policy Learning via Gradient Descent

1: **Inputs:**
2: $S = \{E_1, \ldots, E_N\}$: Training environments
3: $\delta, \delta' \in (0, 1)$: Probability thresholds
4: P_0: Prior over policies
5: $\mu, s \in \mathbb{R}$: Initializations for μ and s
6: γ: step size for gradient descent
7: **Outputs:**
8: $\mu^\star, s^\star$: Optimal μ, s
9: $C^\star_{\text{bound}} := \mathbb{D}^{-1}\left(\bar{C}_S(\mathcal{N}_{\mu^\star,s^\star}; L, \delta') \parallel \frac{\mathbb{D}(\mathcal{N}_{\mu^\star,s^\star} \| P_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{N}\right)$
10: **Procedure:**
11: $B(\mu, s, w) := \frac{1}{N}\sum_{E\in S} C(r_w; E) + \sqrt{\frac{\mathbb{D}(\mathcal{N}_{\mu^\star,s^\star} \| P_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}}$
12: **while** ¬converged **do**
13: Sample $\xi \sim \mathcal{N}_{0,I_d}$ and set $w \leftarrow \mu + \sqrt{s} \odot \xi$
14: $\mu \leftarrow \mu - \gamma \nabla_\mu B(\mu, \exp(\eta), w)$
15: $\eta \leftarrow \eta - \gamma \nabla_\eta B(\mu, \exp(\eta), w)$
16: $s \leftarrow \exp(\eta)$
17: **end while**

$$\hat{C}_S(\mathcal{N}_{\mu^\star,s^\star}) := \frac{1}{NL}\sum_{E\in S}\sum_{i=1}^{L} C(r_{w_i}; E) \tag{22}$$

We can then use a sample convergence bound (see Langford and Caruana, 2002) to bound the error between $\hat{C}_S(\mathcal{N}_{\mu^\star,s^\star})$ and $C_S(\mathcal{N}_{\mu^\star,s^\star})$. In particular, the following bound is an application of the relative entropy version of the Chernoff bound for random variables (i.e., costs) bounded in $[0, 1]$ and holds with probability $1 - \delta'$:

$$\begin{aligned} C_S(\mathcal{N}_{\mu^\star,s^\star}) &\leqslant \bar{C}_S(\mathcal{N}_{\mu^\star,s^\star}; L, \delta') \\ &:= \mathbb{D}^{-1}(\hat{C}_S(\mathcal{N}_{\mu^\star,s^\star}) \parallel \frac{1}{L}\log(\frac{2}{\delta'})) \end{aligned} \tag{23}$$

Combining inequalities (10) and (23) using the union bound, we see that the following bound holds with probability at least $1 - \delta - \delta'$:

$$\begin{aligned} &C_{\mathcal{D}}(\mathcal{N}_{\mu^\star,s^\star}) \leqslant C^\star_{bound} := \mathbb{D}^{-1} \\ &\left(\bar{C}_S(\mathcal{N}_{\mu^\star,s^\star}; L, \delta') \parallel \frac{\mathbb{D}(\mathcal{N}_{\mu^\star,s^\star} \parallel P_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{N}\right) \end{aligned} \tag{24}$$

This is the final version of our bound on the expected performance of policies (drawn from $\mathcal{N}_{\mu^\star,s^\star}$).

Algorithm 2 summarizes our approach from this section. Note that in order to ensure positivity of $s \in \mathbb{R}^d_+$, we perform the optimization with respect to $\eta := \log(s)$.

6. Extensions

In this section, we present two extensions to the basic framework presented so far. In Section 6.1, we discuss extensions to systems with stochastic dynamics or sensor measurements. In Section 6.2, we present an approach that allows us to tackle settings where training and test environments are drawn from different distributions.

6.1. Stochastic rollout functions

In our problem formulation in Section 2, we assumed that the rollout function $r_w : \mathcal{E} \to (\mathcal{X} \times \mathcal{U})^H$ is deterministic (i.e., once the environment is fixed, the resulting state-action trajectory obtained by applying a given policy is completely determined). Here we briefly sketch an extension of our framework to settings where the rollout function is stochastic (e.g., due to stochasticity in the dynamics of the system or in sensor measurements). This is made possible by a reinterpretation of the variable w. Previously, w corresponded to parameters of the control policy. Suppose now that we think of w as consisting of two components $w := [w_{\text{int}}, w_{\text{ext}}]$: an "internal" component w_{int} corresponding to parameters of the control policy (just as before), and an additional "external" component corresponding to uncertain parameters (e.g., external disturbances that the robot might experience). The rollout function now has the following structure: $r_{[w_{\text{int}}, w_{\text{ext}}]} : \mathcal{E} \to (\mathcal{X} \times \mathcal{U})^H$. The stochasticity in w_{int} is directly set by us (i.e., by choosing a prior P_0 and posterior P as before). However, the stochasticity over w_{ext} is beyond our control.

We note that the structure of the resulting problem is identical to the original formulation considered in Section 2. The only difference comes from the fact that a portion of the stochasticity in the rollouts is beyond our control. We can thus apply Theorem 2 directly in order to obtain an upper bound on the true expected cost. In particular, let P_0 and P be the prior and posterior over w_{int} (as before) and suppose that the distribution over w_{ext} is given by P_{ext}. Further, assume that w_{int} and w_{ext} are independent random variables. We can then define $P_{0'}$ and P' to be the prior and posterior distributions over $w := [w_{\text{int}}, w_{\text{ext}}]$ and evaluate the "regularizer" term in the PAC-Bayes bound in Theorem 2 by noting that

$$\mathbb{D}(P' \parallel P_{0'}) = \mathbb{E}_{P,P_{\text{ext}}}\left[\log\left(\frac{P_{\text{ext}}P}{P_{\text{ext}}P_0}\right)\right] = \mathbb{E}_{P,P_{\text{ext}}}\left[\log\left(\frac{P}{P_0}\right)\right] \tag{25}$$

$$= \mathbb{E}_P\left[\log\frac{P}{P_0}\right]\underbrace{\mathbb{E}_{P_{\text{ext}}}[1]}_{=1} = \mathbb{D}(P \parallel P_0) \tag{26}$$

The equality between the two lines follows from the fact that w_{int} and w_{ext} are independent. In order to evaluate the training cost $C_S(P') = \frac{1}{N}\sum_{E\in S}\mathbb{E}_{w\sim P'}[C(r_w; E)]$, we can employ the sampling procedure described in Section 5.2 (i.e., sampling the disturbances $w_{\text{ext}} \sim P_{\text{ext}}$ in a manner analogous to how w was sampled in Section 5.2). Thus, the framework for the deterministic rollout setting can be applied with almost no modifications in order to handle the

stochastic rollout case (as long as one can sample disturbances w_{ext} and assuming that the disturbances are drawn independently of w_{int}).

6.2. Distributionally robust control policies

So far, we have assumed that the robot will be tested on environments that are drawn from the same distribution as the training environments. We now address the setting where this assumption is not valid and learn *distributionally robust policies* (i.e., policies that are robust to changes in the distribution from which environments are drawn). We assume that the distribution $\mathcal{D}'$ from which test environments are drawn is bounded in terms of an *f-divergence* (see the following) from the training distribution $\mathcal{D}$ and formulate a robust version of the PAC-Bayes bound already described.

Definition 1 (*f*-divergence between $\mathcal{D}'$ and $\mathcal{D}$ (Nguyen et al., 2010)) *For any convex $f(x)$ such that $f(1)=0$, let*

$$D_f(\mathcal{D}'||\mathcal{D}) := \mathop{\mathbb{E}}_{E\sim\mathcal{D}}\left[f\left(\frac{\mathcal{D}'}{\mathcal{D}}\right)\right] \tag{27}$$

The f-divergences encapsulate a broad class of divergences between distributions and include the KL divergence as a special case (with $f(x)=x\log x$). We will assume that the test distribution $\mathcal{D}'$ is bounded in terms of an f-divergence: $D_f(\mathcal{D}'||\mathcal{D})\leqslant\mathcal{B}$ (but no further assumption on $\mathcal{D}'$ will be made). The control policy we learn will have an associated guarantee on *any* test distribution that satisfies this assumption.

Theorem 3 (f-divergence between $\mathcal{D}'$ and $\mathcal{D}$ in terms of f and f^*). *For a given $f(x)$ and its convex conjugate $f^*(y) := \sup_{x\in\mathbb{R}}[xy-f(x)]$, we can write the f-divergence $D_f(\mathcal{D}'||\mathcal{D})$ in terms of only f and its conjugate*:

$$D_f(\mathcal{D}'||\mathcal{D}) = \sup_{C:\Pi\times\mathcal{E}\to\mathbb{R}}\left(\mathop{\mathbb{E}}_{E\sim\mathcal{D}'}\mathop{\mathbb{E}}_{w\sim P}[C(r_w;E)] - \mathop{\mathbb{E}}_{E\sim\mathcal{D}}\mathop{\mathbb{E}}_{w\sim P}[f^*(C(r_w;E))]\right) \tag{28}$$

This supremum is taken over all functions C that result in the expectations in the RHS being finite. Thus, for any particular (cost) function C, we obtain a lower bound on the supremum term. This allows us to obtain the following useful corollary.

Corollary 2 (f-divergence variational inequality). *If $D_f(\mathcal{D}'||\mathcal{D})\leqslant\mathcal{B}$, then*

$$C_{\mathcal{D}'}(P) := \mathop{\mathbb{E}}_{E\sim\mathcal{D}'}\mathop{\mathbb{E}}_{w\sim P}[C(r_w;E)] \leqslant \mathcal{B} + \mathop{\mathbb{E}}_{E\sim\mathcal{D}}\mathop{\mathbb{E}}_{w\sim P}[f^*(C(r_w;E))] \tag{29}$$

Note that this corollary is valid for any f-divergence. In particular, it holds when $f(x)=x\log x$, making $f^*(y)=e^{y-1}$. With this choice of f, we obtain an upper bound on $C_{\mathcal{D}'}(P)$ in terms of the bound $\mathcal{B}$ on the KL divergence.

Corollary 3 (KL divergence variational inequality). *For f-divergence with $f(x)=x\log x$ and $D_f(\mathcal{D}'||\mathcal{D})\leqslant\mathcal{B}$, we have $D_f(\mathcal{D}'||\mathcal{D})=\mathbb{D}(\mathcal{D}'||\mathcal{D})\leqslant\mathcal{B}$ and*

$$C_{\mathcal{D}'}(P) \leqslant \mathcal{B} + \mathop{\mathbb{E}}_{E\sim\mathcal{D}}\mathop{\mathbb{E}}_{w\sim P}[e^{C(r_w;E)}] - 1 \tag{30}$$

Whereas Corollary 3 provides a valid inequality in the special case of the KL divergence, a tighter bound can be obtained using the *Donsker–Varadhan* (DV) inequality.

Theorem 4 (DV variational inequality (Donsker and Varadhan, 1975; Gray, 2011, Theorem 3.2)). *If $\mathbb{D}(\mathcal{D}'||\mathcal{D})\leqslant\mathcal{B}$, then*

$$C_{\mathcal{D}'}(P) \leqslant \mathcal{B} + \log\left(\mathop{\mathbb{E}}_{E\sim\mathcal{D}}\mathop{\mathbb{E}}_{w\sim P}[e^{C(r_w;E)}]\right) \tag{31}$$

The DV inequality provides a tighter bound than inequality (30) since $x-1\geqslant\log(x)$, $\forall x>0$. For the rest of this section, we will specialize our discussion to the KL divergence and use the DV inequality. However, we note that our approach generalizes to any f-divergence by leveraging Corollary 2.

As written, inequality (31) cannot be used to directly upper bound $C_{\mathcal{D}'}(P)$ since $\mathbb{E}_{E\sim\mathcal{D}}\,\mathbb{E}_{w\sim P}[e^{C(r_w;E)}]$ is not an observable quantity. However, we can leverage the inequality (14) to obtain an upper bound on $C_{\mathcal{D}'}(P)$ in terms of observable quantities. Since Theorem 2 holds for any cost function between 0 and 1, we will be able to apply inequality (14) if we replace the cost with an exponentiated one, as long as we rescale to stay between 0 and 1. Thus, if we make the substitution

$$C(r_w;E) \leftarrow \frac{e^{C(r_w;E)}-1}{e-1}$$

we obtain the following bound using inequality (14):

$$\mathop{\mathbb{E}}_{E\sim\mathcal{D}}\mathop{\mathbb{E}}_{w\sim P}[e^{C(r_w;E)}] \leqslant \frac{1}{N}\sum_{E\in S}\mathop{\mathbb{E}}_{w\sim P}[e^{C(r_w;E)}] + (e-1)\sqrt{\frac{\mathbb{D}(P\parallel P_0)+\log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}} \tag{32}$$

This inequality holds because the transformation keeps the cost in $[0,1]$. Now, since we assumed that $\mathbb{D}(\mathcal{D}'||\mathcal{D})\leqslant\mathcal{B}$, we can apply Theorem 4 to bound $C_{\mathcal{D}'}(P)$.

Corollary 4 (Distributionally-robust PAC-Bayes bound). *For any $\mathcal{D}'$ such that $\mathbb{D}(\mathcal{D}'||D)\leqslant\mathcal{B}$ and any $\delta\in(0,1)$,*

with probability at least $1-\delta$ *over sampled environments* $S\sim\mathcal{D}^N$ *the following inequality holds*:

$$C_{\mathcal{D}'}(P) \leqslant \mathcal{B} + \log \left(\frac{1}{N} \sum_{E \in S} \mathop{\mathbb{E}}_{w \sim P} [e^{C(r_w;E)}] + (e-1) \sqrt{\frac{\mathbb{D}(P \parallel P_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N}} \right) \tag{33}$$

The RHS of inequality (33) gives us an upper bound $C_{\text{PAC}'}$ on $C_{\mathcal{D}'}$. We can thus apply an analogous procedure to Algorithm 1 to obtain $P^{\star}_{\text{PAC}'}$ (a distributionally robust stochastic policy) by minimizing this upper bound and $C^{\star}_{\text{bound}'}$ (the final distributionally robust PAC-Bayes bound).

In the finite policy space setting, we can apply a procedure similar to that employed in Section 5.1 to write an REP that minimizes the bound $C_{\text{PAC}'}$. Define

$$\hat{C}_e[i,j] e^{C(r_{\pi_j};E_i)} \tag{34}$$

We then have

$$\frac{1}{N} \sum_{E \in S} \mathop{\mathbb{E}}_{\pi \sim P} [e^{C(r_\pi;E)}] = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{L} \hat{C}_e[i,j] p[j] := \bar{C}_e p \tag{35}$$

We can then minimize the bound $C_{\text{PAC}'}$ using an REP analogous to Problem (19):

$$\begin{aligned} \min_{p \in \mathbb{R}^L, \tau, \lambda} \quad & \tau \\ \text{s.t.} \quad & \lambda^2 \geqslant (e-1)^2 \frac{\mathbb{D}(p \parallel p_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{2N} \\ & \lambda = \tau - \bar{C}_e p, \ \lambda \geqslant 0 \\ & 0 \leqslant p \leqslant 1, \ \sum_j p[j] = 1 \end{aligned} \tag{36}$$

Here $\bar{C}_e p \in [1, e]$, and since τ upper bounds the true expected cost, we are only interested in values of $\tau \in [1, e]$. Thus an optimal λ can be found by searching over $\lambda \in [0, e-1]$, which can then be used to obtain $P^{\star}_{\text{PAC}'}$ and $C_{\text{PAC}'}$. In addition, in the continuously parameterized control policy space case, we can make modifications to Algorithm 2 and equations (20–24) to directly adapt the SGD approach to minimize $C_{\text{PAC}'}$.

Finally, as in Section 5, the final distributionally robust upper bound $C^{\star}_{\text{bound}'}$ is not computed as $C_{\text{PAC}'}(P^{\star}_{\text{PAC}'})$ but with an analog to the KL inverse in (11):

$$\begin{aligned} \max_{c_{\mathcal{D}'}, c_{\mathcal{D}} \in [0,1]} \quad & c_{\mathcal{D}'} \\ \text{s.t.} \quad & \mathbb{D}(C_S(P) || c_{\mathcal{D}}) \leqslant \frac{\mathbb{D}(P \parallel P_0) + \log\left(\frac{2\sqrt{N}}{\delta}\right)}{N} \\ & \mathbb{D}(c_{\mathcal{D}'} || c_{\mathcal{D}}) \leqslant \mathcal{B} \end{aligned} \tag{37}$$

The first constraint is the same as in the non-robust case, and the second constraint accounts for the difference in the training and test distributions. Together these create an REP that can be solved to find $C^{\star}_{\text{bound}'}$.

7. Examples

In this section, we demonstrate our framework in simulation on two domains: obstacle avoidance (Section 7.1) and grasping (Section 7.2). Our goal is to demonstrate the ability of our approach to learn control policies with strong guarantees on generalization to novel environments. We consider a hardware example in Section 8.

7.1. Reactive obstacle avoidance control

In this section, we apply our approach on the problem of learning reactive obstacle avoidance policies for a ground vehicle model equipped with a depth sensor. We first consider a finite policy space Π and leverage the REP-based framework described in Section 5.1. We then consider continuously parameterized policies and apply the approach from Section 5.2. Finally, we apply the approach from Section 6.2 to learn distributionally robust control policies.

Dynamics. A pictorial depiction of the ground vehicle model is provided in Figure 1(a). The state of the system is given by $[x, y, \psi]$, where x and y are the x and y positions of the vehicle, respectively, and ψ is the yaw angle. We model the system as a differential drive vehicle with the following nonlinear dynamics:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -\frac{r}{2}(u_l + u_r)\sin(\psi) \\ \frac{r}{2}(u_l + u_r)\cos(\psi) \\ \frac{r}{L}(u_r - u_l) \end{bmatrix} \tag{38}$$

where u_l and u_r are the control inputs (corresponding to the left and right wheel speeds, respectively), $r = 0.1$ m corresponds to the radius of the wheels, and $L = 0.5$ m corresponds to the width of the base of the vehicle. We set

$$u_l = u_0 - u_{\text{diff}}, \quad u_r = u_0 + u_{\text{diff}} \tag{39}$$

where $u_0 = v_0/r$ with $v_0 = 2.5$ m/s. This ensures that the robot has a fixed speed v_0. We limit the turning rate by constraining $u_{\text{diff}} \in [-u_0/2, u_0/2]$. The system is simulated as a discrete-time system with time-step $\Delta t = 0.05$ s.

Obstacle environments. A typical obstacle environment is shown in Figure 1(a) and consists of N_{obs} cylinders of varying radii along with three walls that bound the environment between $x \in [-5, 5]$ m and $y \in [0, 10]$ m. Environments are generated by first sampling the integer N_{obs} uniformly between 20 and 40, and then independently sampling the x–y positions of the cylinders from a uniform distribution over the ranges $x \in [-5, 5]$ m and $y \in [2, 10]$ m. The radius of each obstacle is sampled independently from a uniform distribution over the range $[0.05, 0.2]$ m. The robot's state is always initialized at $[x, y, \psi] = [0, 1, 0]$.

Table 2. Comparison of PAC-Bayes bound with the true expected cost (estimated by sampling 10^5 obstacle environments). Using only 100 samples, with probability 0.99 over samples, the PAC-Bayes policy is guaranteed to have an expected success rate of 82.2%. The true expected success rate is approximately 91.3%.

N (# of training environments)	100	500	1,000	10,000
PAC-Bayes bound ($C^{\star}_{\text{bound}}$)	0.178	0.135	0.121	0.096
True expected cost (estimate)	0.087	0.084	0.088	0.083

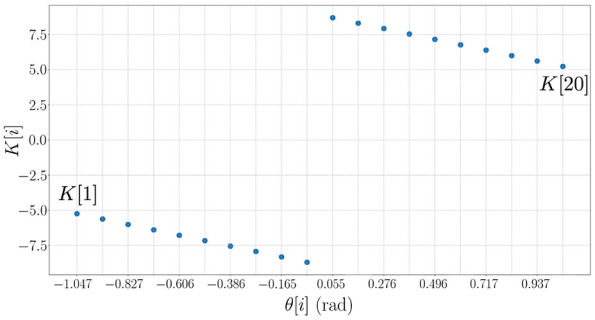


Figure 3. Example of $K[i]$ as a function of $\theta[i]$.

Obstacle avoidance policies. We assume that the robot is equipped with a depth sensor that provides distances $y[i]$ along 20 rays in the range $\theta[i] \in [-\pi/3, \pi/3]$ rad (positive is clockwise) up to a sensing horizon of 5 m (as shown in Figure 1(a)). A given sensor measurement y thus belongs to the space $\mathcal{Y} = \mathbb{R}^{20}$. Let $\hat{y} = 1/y \in \mathbb{R}^{20}$ be the inverse distance vector computed by taking an element-wise reciprocal of y. We then choose u_{diff} as the following dot product:

$$u_{\text{diff}} = K \cdot \hat{y} \tag{40}$$

An example of $K \in \mathbb{R}^{20}$ is

$$K[i] = \begin{cases} (y_0/x_0)(x_0 - \theta[i]) & \text{if } \theta[i] \geqslant 0 \\ (y_0/x_0)(-x_0 - \theta[i]) & \text{if } \theta[i] < 0 \end{cases} \tag{41}$$

Such a K is shown in Figure 3. For $\theta[i] > 0$, $K[i]$ is a linear function of $\theta[i]$ with x- and y-intercepts equal to x_0 and y_0, respectively. This linear function is reflected about the origin for $\theta[i] < 0$.

Intuitively, this corresponds to a simple reactive policy that computes a weighted combination of inverse distances in order to turn away from obstacles that are close. As a simple example, consider the case where we have two obstacles: one located 4 m away along $\theta = -\pi/4$ (i.e., to the robot's left) and the other located 1 m away along $\theta = \pi/4$ (i.e., to the robot's right). The computed control input will then be $u_{\text{diff}} > 0$ (i.e., robot turns left) since the inverse depth for the obstacle to the right is larger than that of the obstacle to the left. Simple reactive policies of this kind have been shown to be quite effective in practice (Arkin, 1998; Beyeler et al., 2009; Conroy et al., 2009; Ross et al., 2013), but can often be challenging to tune by hand in order to achieve good expected performance across *all* environments. We tackle this challenge by applying the PAC-Bayes control framework proposed here.

Results (finite policy space). In order to obtain a finite policy space, we choose $L = 50$ different K of the form (41) by choosing different x and y intercepts x_0 and y_0. In particular, (x_0, y_0) is chosen by discretizing the space $[0.1, 5.0] \times [0, 10.0]$ into 5 values for x_0 and 10 values for y_0. Our control policy space is thus $\Pi = \{\pi_1, \ldots, \pi_L\}$, where each policy π_i corresponds to a particular choice of K.

We consider a time horizon of $T = 100$ and assign a cost of 1 if the robot collides with an obstacle during this period and a cost of 0 otherwise. We choose a uniform prior over the policy space Π and apply the REP framework from Section 5.1 in order to optimize a distribution over policies. The PyBullet package (Coumans and Bai, 2018) is used to simulate the dynamics and depth sensor; we use these simulations to compute the elements of the cost matrix $\bar{C}$ (see Section 5.1). Each simulation takes ~0.01 s to execute in our implementation (note that the computation of the different elements of $\bar{C}$ can be parallelized entirely). Given the matrix $\bar{C}$ with 100 sampled environments, each REP (corresponding to a fixed value of λ in Problem (19)) takes ~0.05 s to solve using the CVXPY package (Diamond and Boyd, 2016) and the SCS solver (O'Donoghue et al., 2017). We discretize the interval $[0, 1]$ into 100 values to find the optimal λ. Complete code for this implementation is freely available on GitHub.[3]

Table 2 presents the upper bound $C^{\star}_{\text{bound}}$ on the true expected cost of the PAC-Bayes control policy $P^{\star}_{\text{PAC}}$ (see Algorithm 1) for different sample sizes N with $\delta = 0.01$. The table also presents an estimate of the true expected cost $C_{\mathcal{D}}(P^{\star}_{\text{PAC}})$ obtained by sampling 10^5 environments. As the table illustrates, the PAC-Bayes bound provides strong guarantees even for relatively small sample sizes. For example, using only 100 samples, the PAC-Bayes policy is guaranteed (with probability $1 - \delta = 0.99$) to have an expected success rate of 82.2% (i.e., an expected cost of 0.178). Exhaustive sampling indicates that the expected success rate for the PAC-Bayes policy is approximately 91.3% for this case. Videos of representative trials on test environments can be found at https://youtu.be/y4zTK79s1mI.

Results (continuous policy space). Next, we consider a continuously parameterized policy space Π and apply the approach described in Section 5.2. In particular, we parameterize our policy using the matrix $K \in \mathbb{R}^{20}$ in equation (40) while ensuring symmetry of the control law, i.e., we

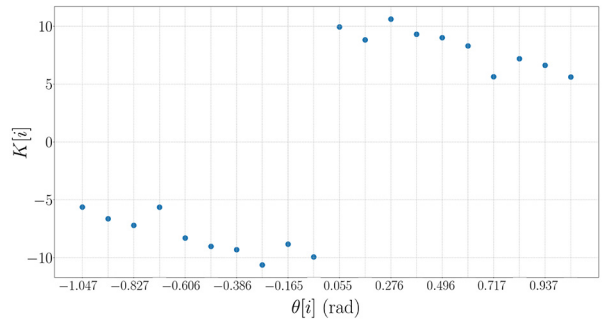


Figure 4. Optimized K corresponding to $\mu^\star$.

constrain $K[i] = -K[j]$ for $\theta[i] = -\theta[j]$ (note that K is no longer constrained to have the linear form from equation (41)). The dimensionality of the parameter space is thus $d = 10$. We apply Algorithm 2 to optimize a distribution $\mathcal{N}_{\mu^\star, s^\star}$ over policies. For the purpose of optimization, we employ a continuous surrogate cost function in place of the discontinuous 0–1 cost. We choose this to be the negative of the minimum distance to an obstacle along a trajectory (appropriately scaled to lie within $[0, 1]$). Note that we employ this surrogate cost only for optimization; all results are presented for the 0-1 cost. Gradients in Algorithm 2 are estimated numerically. We choose a prior $P_0 = \mathcal{N}_{\mu_0, s_0}$ with $s_0 = 0.01$; the mean μ_0 is given by a vector K of the form (41) with x intercept 2.5 and y intercept 10.0.

We use $N = 100$ training environments and choose confidence parameters $\delta = 0.009$, $\delta' = 0.001$, and $L = 30,000$ samples to evaluate the sample convergence bound in (23). Figure 4 shows the mean $\mu^\star$ of the optimized policy obtained using Algorithm 2. The corresponding PAC-Bayes bound $C^\star_{\text{bound}}$ is 0.224. Thus, with probability 0.99 over sampled training data, the optimized PAC-Bayes policy is guaranteed to have an expected success rate of 77.6%. Exhaustive sampling with 10^5 environments indicates that the expected success rate is approximately 92.5%. Videos of representative trials on test environments can be found at https://youtu.be/y4zTK79s1mI.

Results (distributionally robust policies). We now apply the approach presented in Section 6.2 to learn distributionally robust policies. Complete code for the implementation of the example here is freely available on GitHub.[4] To provide a concrete way of bounding $\mathbb{D}(\mathcal{D}'||\mathcal{D})$, the training and test distributions differ only in the way that the radius of the cylindrical obstacles for that environment is sampled. For a single environment, all obstacles will have the same radius, but the beta distribution from which this radius is sampled differ. This means that $\mathbb{D}(\mathcal{D}'||\mathcal{D}) = \mathbb{D}(\mathbb{B}(\alpha', \beta')||\mathbb{B}(\alpha, \beta))$ where $\mathbb{B}(\alpha, \beta)$ is the beta distribution, with parameters α and β, used to sample the radius:

$$\begin{aligned}\mathbb{D}(\mathbb{B}(\alpha', \beta')||\mathbb{B}(\alpha, \beta)) = \log\left(\frac{\mathrm{B}(\alpha, \beta)}{\mathrm{B}(\alpha', \beta')}\right) \\ + (\alpha' - \alpha)\psi(\alpha') + (\beta' - \beta)\psi(\beta') \\ + (\alpha - \alpha' + \beta - \beta')\psi(\alpha' + \beta')\end{aligned} \tag{42}$$

where α and β are the beta distribution parameters for $\mathcal{D}$, α' and β' are the beta distribution parameters for $\mathcal{D}'$, $\mathrm{B}(\cdot, \cdot)$ is the beta function (distinct from the beta distribution), and $\psi(\cdot)$ is the digamma function. This divergence can be computed analytically with a symbolic integrator such as Mathematica (Wolfram Research, Inc., 2019). See Figure 5 for the probability density functions (PDFs) of the distributions used to determine the radii of obstacles for this example, where $\mathbb{D}(\mathcal{D}'||\mathcal{D}) \leqslant \mathcal{B} = 0.0819$. Note that for any test distribution over environments that satisfies the inequality $\mathbb{D}(\mathcal{D}'||\mathcal{D}) \leqslant 0.0819$, the computed bound $C^\star_{\text{bound}'}$ will be valid.

In Figure 6, training and test obstacles are contrasted. Since the training environments are generated with a beta

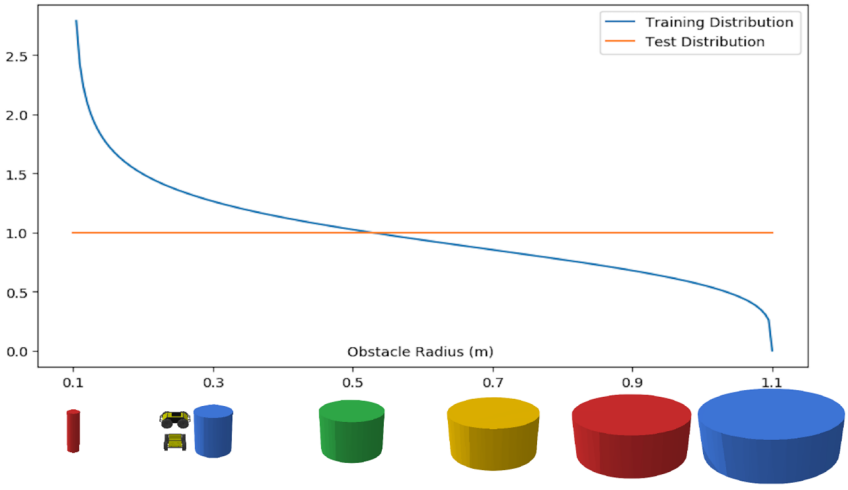


Figure 5. PDFs for the beta distributions used to determine obstacle radii for the training and test environments. Here $\alpha = 0.8$, $\beta = 1.25$, $\alpha' = 1$, and $\beta' = 1$, which makes $\mathbb{D}(\mathcal{D}'||\mathcal{D}) = \mathbb{D}(\mathbb{B}(\alpha', \beta')||\mathbb{B}(\alpha, \beta)) \leqslant \mathcal{B} = 0.0819$. The robot's radius is 0.27 m, depicted next to the 0.3 m obstacle, and the radii of the obstacles are bounded between 0.10 and 1.10 m.

Table 3. Comparison of distributionally robust PAC-Bayes bounds with true costs estimated using 10^5 environments. We are able to obtain strong bounds on generalization (albeit with a larger number of training environments than in the non-robust case). For example, with 5,000 training environments, we obtain a guaranteed expected success rate of 80.3%. The estimated true success rate is approximately 91.8%. We also provide bounds and estimated true costs obtained using the standard (non-robust) PAC-Bayes framework as points of comparison.

N (# of training environments)	100	500	1,000	5,000	10,000
Robust PAC-Bayes bound ($C^{\star}_{\text{bound}'}$)	0.453	0.276	0.238	0.197	0.185
True (estimated) cost on $\mathcal{D}'$ using robust policy learned using $\mathcal{D}$	0.079	0.081	0.080	0.082	0.080
Non-robust PAC-Bayes bound on $\mathcal{D}$	0.221	0.107	0.089	0.070	0.066
True (estimated) cost on $\mathcal{D}$ using policy learned on $\mathcal{D}$	0.054	0.057	0.054	0.054	0.056
Non-robust PAC-Bayes bound on $\mathcal{D}'$	0.262	0.170	0.138	0.110	0.096
True (estimated) cost on $\mathcal{D}'$ using policy learned on $\mathcal{D}'$	0.081	0.080	0.081	0.079	0.083

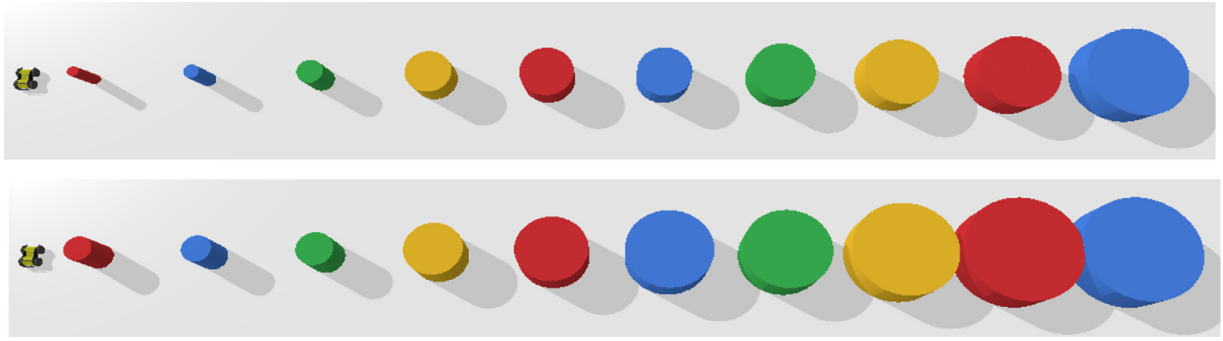


Figure 6. Comparison of obstacles generated by the beta distribution on the radius for the training (top image) and test (bottom image) environments. When the obstacles are sorted, it is easier to see that the training environment's obstacles are skewed towards a smaller radius, making those environments easier to navigate. This is apparent in the PDF comparison displayed in Figure 5 as well.

distribution that favors smaller radii, they are likely to be smaller than those generated with the test beta distribution (uniform distribution over the radius range). In addition, the average obstacle radius, which can be calculated with the beta distribution parameters for the training and test distributions ($r_{\text{avg}} = \alpha/(\alpha+\beta) \times (r_{\max} - r_{\min}) + r_{\min}$), differ by 0.11m. This corresponds to 41% of the robot's radius. Results on this example for the approach presented in Section 6.2 are presented in Table 3. The table demonstrates that we are able to obtain strong bounds on generalization even in this distributionally-robust setting (albeit with a larger number of training environments). For example, with 5000 training environments, we obtain a guaranteed expected success rate of 80.3%. We emphasize that this bound holds for *any* $\mathcal{D}'$ that satisfies the constraint on the KL divergence (not just the specific test distribution chosen here). The estimated true success rate is approximately 91.8% for $N = 5000$. With a sufficiently large number of training environments (10^5 in this example), the robust PAC-Bayes bound $\approx$ true expected cost on test $+\mathcal{B}+$ the scaled regularizer that appears in equation (33).

7.2. Grasping

We now consider the problem of learning neural network-based grasping policies with guarantees on performance across novel objects.

Dynamics and sensors. The system we consider is shown in Figure 1(b) and consists of a KUKA iiwa arm grasping an object placed on a table. The robot is equipped with a camera that provides RGB-D images. The entire simulation (rigid-body dynamics and sensing) is performed using the PyBullet simulator (Coumans and Bai, 2018).

Objects. We use the ShapeNet database (Chang et al., 2015) to generate objects for grasping. ShapeNet consists of more than 50,000 objects and, thus, provides a rich and challenging dataset. We scale the objects so they fit in a 10 cm^3 volume. The masses of the objects are randomly chosen uniformly from the range $[0.05, 0.15]$ kg and the inertia matrices are randomly chosen diagonal matrices with elements chosen uniformly from the range $[0.75, 1.25]$. Objects are initialized in the environment by dropping them from a certain height above the table and allowed to settle. The initial orientation from which they are dropped is also randomized (yaw $\sim\mathcal{N}(0, 0.5^2)$, roll $\sim\mathcal{N}(0, 0.5^2)$, pitch

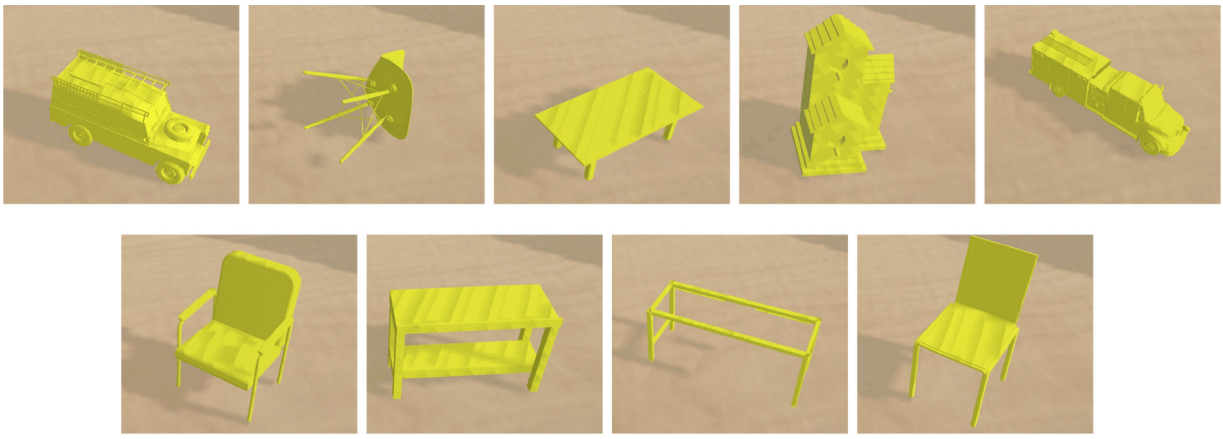


Figure 7. Representative examples of objects from the ShapeNet database.

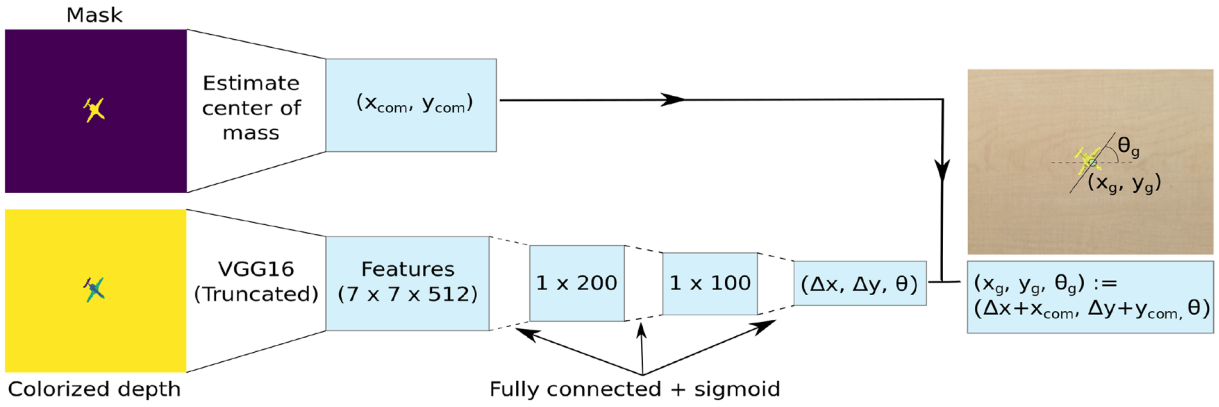


Figure 8. The neural network-based architecture for our grasping policies.

$\sim\mathcal{N}(0, 0.01^2)$). Note that the randomization for the initial pitch angle is smaller; this ensures that objects land "upright" on the table. We randomly select $N = 2,000$ objects as our training data. Figure 7 shows randomly chosen representative objects from the ShapeNet database.

Cost function. We choose a cost function that assigns a cost of 0 if the robot successfully grasps the object and a cost of 1 otherwise. In particular, a "successful" grasp is one that lifts the object to a certain height (>2 cm) above the table.

Neural network policy. Our control policy maps a depth image of an object (and a corresponding mask image) to a grasp location (x_g, y_g) and wrist angle θ_g. A grasp is executed by servoing the robot's gripper to the grasp location (x_g, y_g), setting the wrist angle to the desired angle θ_g, and then executing an open-loop grasping maneuver that closes the grippers and lifts the robot arm up.

The architecture for the pipeline that maps depth images (and corresponding masks) to grasps is illustrated in Figure 8. The mask image is used to estimate the center of mass (COM) $(x_{\text{com}}, y_{\text{com}})$ of the object (by simply computing the centroid of the object). Following Eitel et al. (2015), the raw depth image is *colorized* via a jet colormap. This transforms the depth image from a single-channel image into a three-channel (RGB) image, thus allowing us to re-use neural network architectures pre-trained on the ImageNet dataset (Deng et al., 2009). In particular, we pass the colorized depth image through a VGG16 network (Simonyan and Zisserman, 2014) pretrained on ImageNet and truncated to output a feature representation of size $7 \times 7 \times 512$. This feature vector is passed through three fully connected layers with sigmoid activation. The (distributions over) weights of these fully connected layers (represented in Figure 8 using dashed lines) are learned using the training procedure described in the following. The output of this pipeline is a 1×3 vector $(\Delta x, \Delta y, \theta)$, which is combined with the estimated COM in order to obtain the final target grasp location and orientation $(x_g, y_g, \theta_g) := (\Delta x + x_{\text{com}}, \Delta y + y_{\text{com}}, \theta)$.

Training. We apply the procedure described in Section 5.2 for training our stochastic control policy. In particular, we define Gaussian distributions $\mathcal{N}_{\mu,s}$ over the weights (and biases) of the fully connected layers and choose a Gaussian distribution $\mathcal{N}_{\mu_0,s_0}$ as our prior distribution. Our particular

choice of prior is motivated by the fact that the COM is a reasonable grasp position (i.e., $(\Delta x, \Delta y) = (0, 0)$ is a good guess for the grasp position in the absence of any further knowledge). In particular, we set the prior mean μ_0 by randomly sampling from the distribution $\mathcal{N}_{0,0.001^2}$. Note that while we could have chosen μ_0 to be zero, a randomly chosen μ_0 helps in breaking symmetries in the network (see Dziugaite and Roy (2017a: Appendix B) for a thorough discussion of this point). The prior variance s_0 is set to 0.01.

For the purpose of optimization via SGD, we employ a differentiable surrogate cost function in place of the discontinuous 0–1 cost. In particular, for each object $\mathcal{O}_i$ in our training dataset, we exhaustively attempt 750 grasps by discretizing the space $(\Delta x, \Delta y, \theta) \in [-0.05 \text{ cm}, 0.05 \text{ cm}] \times [-0.05 \text{ cm}, 0.05 \text{ cm}] \times [0 \text{ rad}, \pi \text{ rad}]$ into $5 \times 5 \times 30$ points. As before, $(\Delta x, \Delta y)$ denotes a perturbation from the estimated centroid of the object. For each of the 750 grasps, we record whether the grasp succeeded or failed. We then choose the most "robust" grasp for the given training object by selecting the grasp $(\Delta x^\star, \Delta y^\star, \theta^\star)$ that is most tolerant to errors in θ (i.e., the grasp for which one can perturb θ by the largest magnitude and still successfully grasp the object). This heuristic measure of robustness is motivated by our empirical observation that, in our setting, changes in grasp orientation have a very large effect on whether a grasp succeeds or not, while success is less sensitive to changes in the grasp position. Our surrogate cost function is then computed as the magnitude of the difference between $(\Delta x^\star, \Delta y^\star, \theta^\star)$ and the grasp $(\Delta x, \Delta y, \theta)$ predicted by our neural network policy (note that this difference must take into account the fact that θ lies on a circle and must also be scaled to lie in $[0, 1]$ since costs in our framework are assumed to take values in this range). Importantly, we employ this surrogate cost *only* for optimization; all bounds and results are presented for the 0–1 cost.

Results. We use $N = 2,000$ objects randomly selected from the ShapeNet database as our training objects. We choose confidence parameters $\delta = 0.009, \delta' = 0.001$, and use $L = 1,000$ samples to evaluate the sample convergence bound in equation (23). The resulting PAC-Bayes bound $C^\star_{\text{bound}}$ is 0.294. Thus, with probability 0.99 over sampled training data, the optimized PAC-Bayes control policy is guaranteed to have an expected success rate of 70.6% on novel objects (assuming that they are drawn from the same underlying distribution as the training examples). We hypothesize that this bound could be further improved by using a larger number of samples L in order to evaluate the sample convergence bound in (23) (this would come at an increased computational cost).

We evaluated our PAC-Bayes policy on 1,000 test objects (unseen in the training phase). The policy was successful on 82.0% of these objects. Videos from representative trials on test objects can be found at https://youtu.be/NGI0_oXBdqw.

We also compared our learned policy with a (deterministic) neural network policy trained by minimizing the training cost (i.e., without the regularization that comes from PAC-Bayes). We used an architecture that is identical to the PAC-Bayes policy and initialized weights for the network in the same manner as well (by using the means of the distribution used to define the initialization of the stochastic PAC-Bayes policy). The success rate for the resulting policy on test objects is approximately 78.0% (as compared with 82.0% for the PAC-Bayes policy). We thus see that without the regularization term from PAC-Bayes, the learned policy overfits to a larger degree. We also note that in addition to a loss in the empirical performance, simply minimizing the training cost does not allow us to obtain guarantees on generalization performance.

8. Hardware implementation

In this section, we present results from hardware experiments aimed at validating our approach. The hardware platform we use is the Parrot Swing drone (Figure 2(a)). This lightweight (75 g) quadrotor/fixed-wing hybrid vehicle is an appealing platform since it combines vertical take-off and landing with horizontal flight (thus, making it more efficient than a traditional quadrotor configuration). We implement our approach from Section 5.1 (finite policy spaces) to achieve obstacle avoidance on different environments.

Experimental setup. The Swing takes off from one end of a netted area (see Figure 2(b)) and travels at a fixed speed of 2.0 m/s. To achieve a cost of 0, the Swing must avoid large, cylindrical obstacles over a time horizon of 5 seconds and land safely; otherwise, the Swing will incur a cost of 1. In addition, we consider the net encompassing the area ($7\text{m} \times 18\text{m}$) as "wall" obstacles. We use a Vicon motion tracking system to track the obstacle and Swing's locations. Since the Swing does not possess any sensors for detecting obstacles, we *simulate* a 40-ray depth sensor as if it were mounted on the Swing. This is done using the locations of obstacles, walls, and Swing reported by the motion capture system. Thus, the Swing *only* uses real-time information from this simulated depth sensor; we do not provide any additional information (e.g., the Swing or obstacle locations, etc.). These sensor measurements are provided to a "ground" computer that calculates the control input given a policy. The control inputs to the Swing correspond to percentages of maximum roll, pitch, and yaw angles, as well as the vertical position of the Swing. Commands are sent to the Swing at 10 Hz via Bluetooth using the PyParrot python library (McGovern, 2019). We implement a reactive obstacle avoidance policy (identical to that described in Section 7.1) on the Swing for the discrete policy space setting.

Dynamics model. We train our policies by minimizing the PAC-Bayes bound in simulation; the learned distribution over policies is then implemented on the Swing hardware for validation (on environments not seen during training). We first performed system identification on the Swing in order to obtain an accurate dynamics model. If

we keep the Swing's speed and vertical position constant, we can use a simple model for its dynamics similar to that for the ground vehicle with states $[x, y, \psi]$ (Section 7.1). We can keep the Swing's speed constant by fixing its pitch angle θ. Thus, we fix the vertical position to 1 m, and $\theta = 27°$; the Swing will then travel at about $u_0 = 2.0$ m/s. Consider the following dynamics:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -u_0 \sin(\psi) \\ u_0 \cos(\psi) \\ k_p(k_u u_\psi - \psi) \end{bmatrix} \tag{43}$$

where the only control input u_ψ is a percentage of the Swing's maximum yaw angle ψ, and k_p and k_u are gains. Both k_p and k_u are fit with empirical data to create a realistic simulation. The gain k_u is needed to scale u_ψ such that $k_u u_\psi = \psi$ if u_ψ remains constant and a steady state is reached. We limit $k_u u_\psi \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$ to restrict the Swing to maneuvers that do not significantly change the Swing's forward velocity or vertical position. We first determine k_u by measuring (with the Vicon motion tracking system) the steady-state yaw angle given a fixed control input. We then model the proportional gain k_p with varied input signals such as sinusoidal and chirp functions of varying amplitude. The resulting dynamics, given by $k_u = 3.0$ and $k_p = 0.4$, are implemented in a simulated PyBullet environment analogous to that described for the ground vehicle.

Results. We choose a time horizon $T = 50$; the Swing then flies for 5 s. We then choose $L = 100$ different K in the form of (41) with (x_0, y_0) chosen by discretizing the space $[0.1, 5.0] \times [0, 60.0]$ into 5 and 20 values for x_0 and y_0, respectively. As with the method for the ground vehicle in Section 7.1, we find an upper bound C^*_{bound} on the true expected cost of the PAC-Bayes control policy $P^\star_{\text{PAC}}$ using Algorithm 1. With 1,000 training environments, $P^\star_{\text{PAC}}$ is guaranteed (with 99% probability) to succeed on new environments 88.6% of the time. The empirical success rate, tested on Swing hardware in unseen real-world environments, is approximately 90% (18/20 trials). Videos of representative trials can be found at https://youtu.be/p5CjcSsojg8.

9. Discussion and conclusions

We have presented an approach for learning control policies that provably generalize well to novel environments given a dataset of example environments. Our approach leverages PAC-Bayes theory to obtain upper bounds on the expected cost of (stochastic) policies on novel environments and can be applied to robotic systems with continuous state and action spaces, complicated dynamics, rich sensory inputs, and neural network-based policies. We synthesize policies by explicitly minimizing this upper bound using convex optimization in the case of a finite policy space, and using SGD in the more general case of continuously parameterized policies. We also present an extension of our approach for learning distributionally robust policies, i.e., settings where test environments are drawn from a different distribution than training environments. We demonstrated our framework by learning (i) depth sensor-based obstacle avoidance policies with guarantees on collision-free navigation in novel environments, and (ii) neural network-based grasping policies with guarantees on generalization to new objects. Our simulation results compared the generalization guarantees provided by our technique with exhaustive numerical evaluations in order to demonstrate that our approach is able to provide strong bounds even with relatively few training environments. Our hardware experiments, which tested policies learned using our framework on a real-world obstacle avoidance example, suggest that our technique is effective for developing policies that generalize well to (unseen) real-world environments. We believe that taken together, the simulation and hardware results provide significant evidence for the ability of our approach to provide strong generalization guarantees in realistic robot control settings.

9.1. Challenges and future work

There are a number of challenges and exciting opportunities for future work on both the theoretical and practical fronts. We highlight a few such directions here.

Deterministic policies. It may be desirable in many cases (e.g., safety-critical settings) to learn deterministic policies instead of stochastic ones. Techniques for converting stochastic hypotheses into deterministic hypotheses have been developed within the PAC-Bayes framework (e.g., using majority voting in the classification setting (Lacasse et al., 2007; Langford and Shawe-Taylor, 2003)); an interesting avenue for future work is to extend such techniques to the policy learning setting we consider here. Another possibility is to use different frameworks for obtaining generalization bounds that are better suited to deterministic policies (e.g., bounds based on algorithmic stability (Bousquet and Elisseeff, 2002; Hardt et al., 2015; Kearns and Ron, 1999) and sample compression (Floyd and Warmuth, 1995; Langford, 2005)). An important feature of the reduction-based perspective we presented in Section 4 is that it immediately allows us to port over such bounds from the supervised learning setting to our setting.

Choosing the prior. While we have demonstrated that our framework allows us to obtain strong bounds on generalization performance, an important direction for future work is to find ways to further improve these bounds. We believe that a particularly promising approach for doing this is to systematically choose the prior P_0 over the control policy space. The ability to specify a strong prior is an important distinction between the robot control settings considered in this article and standard supervised learning problems (e.g., image recognition). For standard supervised learning problems, it is often challenging to specify a prior over the space of hypotheses. While the priors in the examples considered in this paper were chosen in a fairly simplistic manner (e.g., a uniform prior over the finite policy space for

the obstacle avoidance example in Section 7.1, or a prior that attempts to keep the grasp position close to the COM in the grasping example in Section 7.2), we believe that choosing priors in a more systematic manner could significantly improve the generalization bounds. One possibility for choosing a prior more carefully is to embed domain knowledge into the prior; for example, one could choose a prior that incorporates a physics model of the system, or one that is derived from an existing state-of-the-art approach for the problem under consideration (e.g., choosing a prior that encourages force-closure grasps). Another promising possibility is to learn the prior from a human expert using imitation learning. By incorporating such priors for robot control problems, we may need significantly smaller datasets than those currently used to train state-of-the-art supervised learning models while still obtaining strong generalization guarantees.

Incorporating different regularizers. The algorithmic approach we employ in this work (Section 5) involves minimizing a combination of the training cost and a regularizer specified by PAC-Bayes theory. This is motivated by the desire to optimize the PAC-Bayes upper bound on the expected cost on novel environments. However, we note that there are a variety of regularization techniques that have been empirically demonstrated to promote generalization (e.g., dropout (Srivastava et al., 2014) and overparameterization (Arora et al., 2018; Neyshabur et al., 2014; Zhang et al., 2016)), in addition to other techniques such as domain randomization (Tobin et al., 2017) and batch normalization (Ioffe and Szegedy, 2015). While these techniques do not yet have strong generalization bounds associated with them, there is a growing body of literature on this topic (Arora et al., 2018; Bjorck et al., 2018; Li and Liang, 2018; McAllester, 2013). Incorporating different regularization schemes into our framework while maintaining strong generalization guarantees is a promising direction for future work.

Extensions to meta-learning. Another exciting future direction is to combine the techniques presented here with *meta-learning* techniques in order to achieve provably data-efficient control on novel tasks. Specifically, we are currently investigating using a PAC-Bayes bound as part of the objective of a meta-learning algorithm such as MAML (Finn et al., 2017) to achieve improved generalization performance and few-shot learning.

We believe that the approach presented here along with the indicated future directions represent an important step towards learning control policies with provable guarantees for challenging robotic platforms with rich sensory inputs operating in novel environments.

Acknowledgments

The authors are grateful to Max Goldstein for initiating the grasping example in Section 7.2 and contributions to the conference version of this article presented at CoRL 2018. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The authors were partially supported by the Office of Naval Research (award number N00014-18-1-2873), the National Science Foundation (grant number IIS-1755038), the Google Faculty Research Award, and the Amazon Research Award.

Notes

1 Note that we are considering a slightly restricted form of the supervised learning problem where each input $z \in \mathcal{Z}$ has only one correct label $z' \in \mathcal{Z}'$. The loss, thus, only depends on the input z and the label $h_w(z)$. The PAC-Bayes framework applies to the more general setting where there is an underlying true distribution on $\mathcal{Z} \times \mathcal{Z}'$ and the loss, thus, has the form $l : \mathcal{H} \times \mathcal{Z} \times \mathcal{Z}' \to \mathbb{R}$. However, the more restricted setting is sufficient for our needs here.

2 The bound we state here is due to Maurer (Maurer, 2004) and improves slightly upon the original PAC-Bayes bounds (McAllester, 1999). The stated bound holds when costs are bounded in the range $[0, 1]$ (as assumed here) and we have $N \geqslant 8$ samples.

3 See https://github.com/irom-lab/PAC-Bayes-Control

4 See https://github.com/irom-lab/PAC-Bayes-Control/tree/master/Extension-Domain_Shifts

References

Agrawal P, Nair AV, Abbeel P, Malik J and Levine S (2016) Learning to poke by poking: Experiential learning of intuitive physics. In: *Advances in Neural Information Processing Systems*, pp. 5074–5082.

Althoff D, Althoff M and Scherer S (2015) Online safety verification of trajectories for unmanned flight with offline computed robust invariant sets. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3470–3477.

Arkin RC (1998) *Behavior-Based Robotics*. Cambridge, MA: MIT Press.

Arora S, Cohen N and Hazan E (2018) On the optimization of deep networks: Implicit acceleration by overparameterization. *arXiv preprint arXiv:1802.06509*.

Bagnell JA (2004) *Learning Decisions: Robustness, Uncertainty, and Approximation*. PhD Thesis, Carnegie Mellon University, Pittsburgh, PA.

Bagnell JA and Schneider JG (2001) Autonomous helicopter control using reinforcement learning policy search methods. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2. IEEE, pp. 1615–1620.

Beyeler A, Zufferey JC and Floreano D (2009) Vision-based control of near-obstacle flight. *Autonomous Robots* 27(3): 201.

Bjorck N, Gomes CP, Selman B and Weinberger KQ (2018) Understanding batch normalization. In: *Advances in Neural Information Processing Systems*, pp. 7694–7705.

Blackmore L, Li H and Williams B (2006) A probabilistic approach to optimal robust path planning with obstacles. In: *Proceedings of the IEEE American Control Conference (ACC)*. IEEE.

Bousquet O and Elisseeff A (2002) Stability and generalization. *Journal of Machine Learning Research* 2: 499–526.

Boyd S and Vandenberghe L (2004) *Convex Optimization*. Cambridge: Cambridge University Press.

Brafman RI and Tennenholtz M (2002) R-max – A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3(Oct): 213–231.

Chandrasekaran V and Shah P (2017) Relative entropy optimization and its applications. *Mathematical Programming* 161(1-2): 1–32.

Chang AX, Funkhouser T, Guibas L, et al. (2015) *ShapeNet: An information-rich 3D model repository.* Technical Report, Stanford University–Princeton University–Toyota Technological Institute at Chicago.

Charnes A and Cooper WW (1959) Chance-constrained programming. *Management Science* 6(1): 73–79.

Conroy J, Gremillion G, Ranganathan B and Humbert JS (2009) Implementation of wide-field integration of optic flow for autonomous quadrotor navigation. *Autonomous Robots* 27(3): 189.

Coumans E and Bai Y (2018) PyBullet, a Python module for physics simulation for games, robotics and machine learning. Available at: https://pybullet.org/wordpress/

Deng J, Dong W, Socher R, Li LJ, Li K and Fei-Fei L (2009) ImageNet: A large-scale hierarchical image database. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 248–255.

Diamond S and Boyd S (2016) CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research* 17(83): 1–5.

Domahidi A, Chu E and Boyd S (2013) ECOS: An SOCP solver for embedded systems. In: *European Control Conference (ECC)*, pp. 3071–3076.

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

Dziugaite GK and Roy DM (2017a) Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*.

Dziugaite GK and Roy DM (2017b) Entropy-SGD optimizes the prior of a PAC-Bayes bound: Data-dependent PAC-Bayes priors via differential privacy. *arXiv preprint arXiv:1712.09376*.

Eitel A, Springenberg JT, Spinello L, Riedmiller M and Burgard W (2015) Multimodal deep learning for robust RGB-D object recognition. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 681–687.

Fard MM and Pineau J (2010) PAC-Bayesian model selection for reinforcement learning. In: *Advances in Neural Information Processing Systems*, pp. 1624–1632.

Fard MM, Pineau J and Szepesvári C (2012) PAC-Bayesian policy evaluation for reinforcement learning. *arXiv preprint arXiv:1202.3717*.

Finn C, Abbeel P and Levine S (2017) Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400* .

Floyd S and Warmuth M (1995) Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning* 21(3): 269–304.

Fraichard T (2007) A short paper about motion safety. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1140–1145.

Francis BA (1987) *A Course in H-Infinity Control Theory.* Berlin: Springer-Verlag.

Fu J and Topcu U (2014) Probably approximately correct MDP learning and control with temporal logic constraints. *arXiv preprint arXiv:1404.7073*.

Germain P, Lacasse A, Laviolette F and Marchand M (2009) PAC-Bayesian learning of linear classifiers. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. New York: ACM Press, pp. 353–360.

Gray RM (2011) *Entropy and Information Theory.* 2nd Ed. New York: Springer Science & Business Media.

Gupta S, Davidson J, Levine S, Sukthankar R and Malik J (2017a) Cognitive mapping and planning for visual navigation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2616–2625.

Gupta S, Fouhey D, Levine S and Malik J (2017 b) Unifying map and landmark based representations for visual navigation. *arXiv preprint arXiv:1712.08125*.

Hardt M, Recht B and Singer Y (2015) Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*.

Ioffe S and Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Kaelbling LP, Littman ML and Cassandra AR (1998) Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2): 99–134.

Kearns M and Ron D (1999) Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation* 11(6): 1427–1453.

Kearns M and Singh S (2002) Near-optimal reinforcement learning in polynomial time. *Machine Learning* 49(2–3): 209–232.

Kearns MJ, Mansour Y and Ng AY (2000) Approximate planning in large POMDPs via reusable trajectories. In: *Advances in Neural Information Processing Systems*, pp. 1001–1007.

Lacasse A, Laviolette F, Marchand M, Germain P and Usunier N (2007) PAC-Bayes bounds for the risk of the majority vote and the variance of the Gibbs classifier. In: *Advances in Neural Information Processing Systems*, pp. 769–776.

Langford J (2005) Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research* 6: 273–306.

Langford J and Caruana R (2002) (Not) bounding the true error. In: *Advances in Neural Information Processing Systems*, pp. 809–816.

Langford J and Shawe-Taylor J (2003) PAC-Bayes and margins. In: *Advances in Neural Information Processing Systems*, pp. 439–446.

Lenz I, Lee H and Saxena A (2015) Deep learning for detecting robotic grasps. *The International Journal of Robotics Research* 34(4–5): 705–724.

Levine S, Finn C, Darrell T and Abbeel P (2016) End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17(1): 1334–1373.

Li Y and Liang Y (2018) Learning overparameterized neural networks via stochastic gradient descent on structured data. In: *Advances in Neural Information Processing Systems*, pp. 8157–8166.

Mahler J, Liang J, Niyaz S, et al. (2017) Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*.

Majumdar A and Goldstein M (2018) PAC-Bayes Control: synthesizing controllers that provably generalize to novel environments. In: *Proceedings of the Conference on Robot Learning (CoRL)*.

Majumdar A and Tedrake R (2017) Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research* 36(8): 947–982.

Maurer A (2004) A note on the PAC Bayesian theorem. *arXiv preprint cs/0411099*.

McAllester D (2013) A PAC-Bayesian tutorial with a dropout bound. *arXiv preprint arXiv:1307.2118*.

McAllester DA (1999) Some PAC-Bayesian theorems. *Machine Learning* 37(3): 355–363.

McGovern A (2019) PyParrot 1.5.*21*. https://github.com/amymcgovern/pyparrot.

MOSEK ApS (2019) Mosek fusion API for Python 9.0.84(beta). https://docs.mosek.com/9.0/pythonfusion/index.html.

Nesterov Y and Nemirovskii A (1994) *Interior-point Polynomial Algorithms in Convex Programming*. Philadelphia, PA: SIAM.

Neu G, Jonsson A and Gómez V (2017) A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*.

Neyshabur B, Bhojanapalli S, McAllester D and Srebro N (2017a) Exploring generalization in deep learning. In: *Advances in Neural Information Processing Systems*, pp. 5949–5958.

Neyshabur B, Bhojanapalli S, McAllester D and Srebro N (2017b) A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. *preprint arXiv:1707.09564*.

Neyshabur B, Tomioka R and Srebro N (2014) In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*.

Nguyen X, Wainwright MJ and Jordan MI (2010) Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory* 56(11): 5847–5861.

O'Donoghue B, Chu E, Parikh N and Boyd S (2016) Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications* 169(3): 1042–1068.

O'Donoghue B, Chu E, Parikh N and Boyd S (2017) SCS: Splitting conic solver, version 2.0.*2*. https://github.com/cvxgrp/scs.

Ono M, Pavone M, Kuwata Y and Balaram J (2015) Chance-constrained dynamic programming with application to risk-aware robotic space exploration. *Autonomous Robots* 39(4): 555–571.

Richter C and Roy N (2017) Safe visual navigation via deep learning and novelty detection. In: *Proceedings of Robotics: Science and Systems (RSS)*.

Richter C, Vega-Brown W and Roy N (2015) Bayesian learning for safe high-speed navigation in unknown environments. In: *Proceedings of the International Symposium on Robotics Research (ISRR)*.

Ross S, Melik-Barkhudarov N, Shankar KS, et al. (2013) Learning monocular reactive UAV control in cluttered natural environments. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1765–1772.

Schouwenaars T, How J and Feron E (2004) Receding horizon path planning with implicit safety guarantees. In: *Proceedings of the IEEE American Control Conference (ACC)*, Vol. 6. IEEE, pp. 5576–5581.

Schulman J, Levine S, Abbeel P, Jordan M and Moritz P (2015) Trust region policy optimization. In: *Proceedings of the International Conference on Machine Learning*. pp. 1889–1897.

Seeger M (2002) PAC-Bayesian generalisation error bounds for Gaussian process classification. *Journal of Machine Learning Research* 3(Oct): 233–269.

Simonyan K and Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556.

Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R (2014) Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1): 1929–1958.

Sünderhauf N, Brock O, Scheirer W, et al. (2018) The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research* 37(4–5): 405–420.

Tobin J, Zaremba W and Abbeel P (2017) Domain randomization and generative models for robotic grasping. *arXiv preprint arXiv:1710.06425*.

Vitus MP and Tomlin CJ (2011) Closed-loop belief space planning for linear, Gaussian systems. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2152–2159.

Wolfram, Research Inc. (2019) Mathematica, Version 12.0. https://www.wolfram.com/mathematica/.

Zhang C, Bengio S, Hardt M, Recht B and Vinyals O (2016) Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

Zhu Y, Mottaghi R, Kolve E, et al. (2017) Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3357–3364.