Projections for Approximate Policy Iteration Algorithms

Riad Akrour [1] **Joni Pajarinen** [1 2] **Gerhard Neumann** [3 4] **Jan Peters** [1 5]

Abstract

Approximate policy iteration is a class of reinforcement learning (RL) algorithms where the policy is encoded using a function approximator and which has been especially prominent in RL with continuous action spaces. In this class of RL algorithms, ensuring increase of the policy return during policy update often requires to constrain the change in action distribution. Several approximations exist in the literature to solve this constrained policy update problem. In this paper, we propose to improve over such solutions by introducing a set of projections that transform the constrained problem into an unconstrained one which is then solved by standard gradient descent. Using these projections, we empirically demonstrate that our approach can improve the policy update solution and the control over exploration of existing approximate policy iteration algorithms.

1. Introduction

Reinforcement learning (RL) formulates a general machine learning problem in which an agent has to take a sequence of decisions to maximize a supervision signal (Sutton & Barto, 1998; Szepesvari, 2010). Because the agent's decisions influence the data gathering process, RL violates the typical assumption of other machine learning settings that data is independent and identically distributed (Bishop, 2006). To cope with this challenge, several RL algorithms constrain the agent's behavior to only slowly change. In trajectory optimization and optimal control, a new policy is made close to the policy around which the dynamics of the system have been approximated through mixing (Todorov & L., 2005; Tassa et al., 2014) or by a Kullback-Leibler (KL) constraint (Levine & Abbeel, 2014b). In policy gradient, a key breakthrough was the use of natural gradient that follows the steepest descent in behavioral space rather than parameter space (Bagnell & Schneider, 2003; Peters & Schaal, 2008; Bhatnagar et al., 2009), i.e. seeking maximal objective improvement with minimal *behavioral* change.

Constraining successive policies to be close to each other in approximate policy iteration is justified in the seminal work of Kakade & Langford 2002 by the mismatch between what the policy update should optimize and what is optimized in practice. As in optimal control closeness can be achieved by mixing policies (Kakade & Langford, 2002; Pirotta et al., 2013), limiting deviation of their probability ratio to one (Schulman et al., 2017) or constraining their KL (Schulman et al., 2015; Abdolmaleki et al., 2018; Tangkaratt et al., 2018; Akrour et al., 2018). To solve KL constrained policy updates, Schulman et al. 2015 use a quadratic approximation of the KL which augments natural policy gradient algorithms with a line-search step critically ensuring KL constraint satisfaction; while Abdolmaleki et al. 2018, Tangkaratt et al. 2018 and Akrour et al. 2018 rely on the method of Lagrange multipliers to derive a general solution in closed form before using sample-based approximations to fit the policy to this solution.

We propose in this paper an alternative approach to policy update under KL and entropy constraints. The core of our approach lies in deriving a projection g mapping the parametric policy space to a subspace thereof complying with the constraints. The constrained maximization of the policy update objective f is then solved by an unconstrained maximization of $f \circ g$. The projections derived in this paper are independent of f and are shown to be relevant in several scenarios pertaining to RL such as direct policy search and approximate policy iteration, respectively introduced in Sec. 2.1 and 2.2. We derive projections for entropy and KL constraints of Gaussian search distributions in Sec. 3.1 before extending them to state-conditioned policies in Sec. 3.2. In the experimental section we show for direct policy search that our more direct approach to constrained optimization sidesteps the need for sample based approximations, yielding an algorithm more robust to low sample counts. We additionally show that the same optimization scheme results in large performance gains by improving the policy update of two existing approximate policy iteration algorithms.

[1]IAS, TU Darmstadt, Darmstadt, Germany [2]Tampere University, Finland [3]L-CAS, University of Lincoln, Lincoln, United Kingdom [4]Bosch Center for Artificial Intelligence (BCAI), Germany [5]Max Planck Institute for Intelligent Systems, Tübingen, Germany. Correspondence to: Riad Akrour <riad@robot-learning.de>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

2. Problem definition

We briefly introduce Direct Policy Search (DPS) and Approximate Policy Iteration (API) and two optimization approaches from the literature to tackle each setting. The optimization problem for API is a straightforward extension of the DPS problem to state conditioned distributions. Similarly, the projections derived in Sec. 3.2 for API will be an extension of those derived for DPS but are more easily understood in the context of the simpler, state-independent, distributions of DPS.

2.1. Direct policy search

Among the wide variety of approaches in policy search (Deisenroth et al., 2013), a distinguishing property of DPS is its reliance on parameter-space exploration as opposed to action space exploration. In parameter-space exploration, a search distribution samples parameters of deterministic policies. In contrast, action space exploration adds noise to every time-step. We refer the reader to Deisenroth et al. 2013, Sec. 2.1, for a more in-depth discussion on exploration strategies in policy search. DPS has applications in robotics for its less jerky exploration scheme, causing less wear and tear to the robot. When combined with specialized low dimensional policies, DPS can solve complex tasks in a model-free fashion, running directly on robotic platforms (Parisi et al., 2015). In simulation, parameter exploration was also used to train larger, neural network based, policies (Plappert et al., 2017; Conti et al., 2018).

Formally, DPS seeks a policy parameter maximizing a noisy reward signal R. To this end it maintains a search distribution, usually of Gaussian shape (Heidrich-Meisner & Igel, 2009; da Silva et al., 2012; Abdolmaleki et al., 2015), $\pi = \mathcal{N}(\mu, \Sigma)$. From π, policy parameters are sampled and evaluated and π is iteratively updated. We will focus on a simple and well founded formulation of DPS that maximizes an expected reward objective L under a KL constraint between successive search distributions. The KL constraint is akin to specifying a learning rate, trading-off exploration and exploitation and preventing the search distribution from collapsing to a point-mass after a single iteration.

Search distribution update optimization problem. At each iteration having sampled and evaluated K parameters from a search distribution q, the algorithm updates q by solving the following constrained optimization problem

$$\begin{aligned} \arg\max_{\pi} \quad & L(\pi), && (1)\\ \text{subject to} \quad & \mathrm{KL}(\pi \parallel q) \leq \epsilon, && (2)\\ & \mathcal{H}(\pi) \quad \geq \beta, && (3) \end{aligned}$$

where $\epsilon \in \mathbb{R}^+$, $\beta \in \mathbb{R}$ and $\mathcal{H}$ denotes the entropy of a distribution. $L(\pi)$ is approximated using importance sampling and the K parameters sampled from q yielding

$$L(\pi) \approx \frac{1}{K} \sum_{\theta^{[i]} \sim q} \frac{\pi(\theta^{[i]})}{q(\theta^{[i]})} R^{[i]}(\theta^{[i]}). \quad (4)$$

The use of importance sampling in Eq. (4) behaves well in practice because constraint (2) enforces π and q to be close to each other.

This problem is identical to the one solved by MORE (Abdolmaleki et al., 2015) which adds to the parameter exploration version of REPS (Deisenroth et al. 2013, Sec. 2.4.3) an entropy constraint given by (3)[1]. The problem solved by MORE is sufficiently general to have other applications in e.g. variational inference (Arenz et al., 2018) or Bayesian optimization (Akrour et al., 2017). Using the method of Lagrange multipliers, one can solve the aforementioned optimization problem in closed form but the resulting distribution is not necessarily Gaussian and requires an additional sample-based approximation to yield a Gaussian π—as further discussed in App. B. To avoid an additional sample-based approximation step, we derive in Sec. 3.1 projections that will ensure compliance with constraints (2) and (3), allowing for a more direct, gradient-based, approach to this optimization problem.

2.2. Approximate policy iteration

The constrained optimization problems of API (Bertsekas, 2011; Scherrer, 2014) and DPS can be surprisingly close, although their problem constraints have different justifications in the literature. To formalize the API policy update problem we make use of "MDPNv1", that is, a standard notation of Markov Decision Processes (MDPs) defined in (Thomas & Okal, 2015). Additionally, for policy π we define the Q-function $Q_\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a\right]$, where the expectation is taken w.r.t. random variables s_t and a_t for $t > 0$; the value function $V_\pi(s) = \mathbb{E}_{a\sim\pi(.|s)}\left[Q_\pi(s,a)\right]$ and the advantage function $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$. The goal in API is to find the policy maximizing the policy return $J(\pi) = V_\pi(s_0)$ for some starting state s_0. In the following we abuse the notation $s \sim \pi$ to mean sampling from the (γ-discounted) state distribution associated with the execution of π.

Policy update optimization problem. Once the advantage function of current policy q is estimated, the policy update in API seeks π such that $J(\pi) > J(q)$. The policy update in API usually proceeds by maximizing

[1] The practical interest of the entropy constraint is to allow the mean and covariance of the search distribution to update at different rates.

$\hat{L}(\pi; q) = \mathbb{E}_{s\sim q}\left[\mathbb{E}_{a\sim\pi}\left[A_q(s,a)\right]\right]$, as described in e.g. Kakade & Langford 2002; Pirotta et al. 2013. $\hat{L}(\pi; q)$ is a proxy to the more direct maximization of $L(\pi; q) = \mathbb{E}_{s\sim\pi}\left[\mathbb{E}_{a\sim\pi}\left[A_q(s,a)\right]\right]$ where the leftmost expectation is w.r.t. the state distribution of π instead of q. It is a more direct objective since $L(\pi; q) = (1-\gamma)\left(J(\pi) - J(q)\right)$ as shown in Kakade & Langford 2002, Lemma 6.1, and hence maximizing L is equivalent to maximizing J. Unfortunately, L is significantly more expensive to evaluate since state samples from π are not available. Because of this discrepancy in state distribution, an unconstrained maximization of $\hat{L}$ may yield a policy with a worse policy return than that of q.

On the other hand, if π and q are close enough so will their state distribution and Kakade & Langford 2002 showed that a positive $\hat{L}$ will imply a positive L in this case. Closeness between π and q was enforced in prior work by mixing the greedy update policy with q (Kakade & Langford, 2002; Pirotta et al., 2013) or by bounding the KL divergence between π and q (Peters et al., 2010; Levine & Abbeel, 2014a; Schulman et al., 2015; Achiam et al., 2017). It is the latter approach that we will consider, with the following optimization problem

$$\begin{aligned} \arg\max_{\pi} \quad & \hat{L}(\pi; q), && (5)\\ \text{subject to} \quad & \mathbb{E}_{s\sim q}\left[\mathrm{KL}(\pi(.|s) \parallel q(.|s))\right] \leq \epsilon, && (6)\\ & \mathbb{E}_{s\sim q}\left[\mathcal{H}(\pi(.|s))\right] \geq \beta, && (7) \end{aligned}$$

Compared to the DPS optimization problem introduced earlier, the optimization here carries over a state-conditioned distribution and the KL and entropy constraints are expressed in average of the state distribution. Handling these expectations will be the main difference between projections for DPS and API.

3. Constraint projections

We derive in this section projections—i.e. idempotent mappings—of the parameteric policy[2] space to a subspace thereof complying with the update constraints introduced in the previous section. Let $\mathcal{F} \subseteq \mathbb{R}^n$ be the subset of policy parameters complying with the constraints in either the DPS or API case. In the remainder of this section we will introduce several projections transforming a constrained optimization problem to an unconstrained one. To show equivalence of the optimization problems, for each projection g we will show that g is a surjective mapping from $\mathbb{R}^n$ to $\mathcal{F}$. That is, g only returns parameters in $\mathcal{F}$, and covers all of $\mathcal{F}$.

Projections for Gaussian policies in the DPS case are derived in Sec. 3.1 before being extended in Sec. 3.2 to state-conditioned Gaussian distributions for API. For clarity of exposition we focus in this paper on Gaussian policies and discuss projections for discrete action spaces in App. A.

[2]We use policy to also refer to DPS's search distribution.

3.1. Direct policy search

To solve the optimization problem defined in Sec. 2.1 we will use a series of projections that will ensure that all parameterizations of a search distribution comply with constraints (2) and (3). To fix ideas, let us first consider an entropy equality constraint of a diagonal covariance matrix where the inequality in (3) is replaced with an equality.

Let $\pi = \mathcal{N}(\mu, \Sigma)$ be a Gaussian with diagonal covariance Σ. We recall that the entropy of a Gaussian distribution only depends on its covariance matrix and the notation $\mathcal{H}(\Sigma)$ will be used interchangeably with $\mathcal{H}(\pi)$ and is given by $\mathcal{H}(\Sigma) = \frac{1}{2}\log(|2\pi e\Sigma|)$. Finally we define

$$h(\lambda, c) = \left(\frac{d}{2}\log(2\pi e) + \sum_i \lambda_i\right) - \beta, \qquad (8)$$

where the inner most term is the entropy of some diagonal covariance matrix having vector $\exp(2\lambda) \in \mathbb{R}^d$ in its diagonal and β is the target entropy. The first parameterization that transforms a constrained problem to an unconstrained one is given by the following property.

Proposition 1. *Optimizing any function $L(\pi)$ w.r.t. mean vector μ and diagonal matrix Σ of a Gaussian $\pi = \mathcal{N}(\mu, \Sigma)$ under entropy equality constraint $\mathcal{H}(\pi) = \beta$ is equivalent to the unconstrained optimization of $L(\pi)$ w.r.t. mean vector μ and the real valued parameter vector λ such that $\Sigma_{i,i} = \exp^2(\lambda_i - \frac{1}{d}h(\lambda, \beta))$ with h as defined in Eq. (8).*

Proof sketch. Through straightforward computations using the definition of h, the Gaussian with diagonal covariance as given in Prop. 1 has entropy of exactly β. Conversely, if $\mathcal{H}(\Sigma) = \beta$, setting $\lambda_i = \frac{1}{2}\log(\Sigma_{i,i})$ will yield back Σ since $h(\lambda, \beta) = 0$. Hence optimizing $L(\pi)$ w.r.t. Σ under constraint $\mathcal{H}(\pi) = \beta$ is equivalent to the unconstrained optimization of $L(\pi)$ w.r.t. λ. □

Prop. 1 defines a projection g that maps any diagonal covariance matrix to a diagonal covariance matrix having an entropy of exactly β by rescaling it with $\exp(\frac{2}{d}h(\lambda, \beta)))$. As this projection is differentiable, and assuming L is also differentiable—which is true for Eq. (4)—one can use gradient ascent for the unconstrained maximization of $L \circ g$. In the following, we propose differentiable projections (differentiable at least outside of the constraint boundary) to the inequality constraint $\mathcal{H}(\pi) \geq \beta$, to full covariance matrices and to the KL constraint.

Proposition 2. *Optimizing any function $L(\pi)$ w.r.t. mean vector μ and diagonal matrix Σ of a Gaussian $\pi = \mathcal{N}(\mu, \Sigma)$, under entropy inequality constraint $\mathcal{H}(\pi) \geq \beta$ is equivalent*

Algorithm 1 DPS Gaussian policy projection

Input: μ, λ, $\lambda_{\text{off\_diag}}$, $q = \mathcal{N}(\mu_q, \Sigma_q)$, ϵ and β
Output: $\pi = \mathcal{N}(\mu, \Sigma)$ complying with KL (2) and entropy (3) constraints
Σ = Entropy_projection($\lambda, \lambda_{\text{off\_diag}}, \beta$)
if $\mathrm{KL}(\mathcal{N}(\mu, \Sigma) \parallel q) > \epsilon$ **then**
  $\eta_g = \frac{\epsilon}{m_q(\mu)+r_q(\Sigma)+e_q(\Sigma)}$
  $\Sigma = \eta_g \Sigma + (1-\eta_g)\Sigma_q$
end if
if $\mathrm{KL}(\mathcal{N}(\mu, \Sigma) \parallel q) > \epsilon$ **then**
  $\eta_m = \sqrt{\frac{\epsilon - r_q(\Sigma) - e_q(\Sigma)}{m_q(\mu)}}$
  $\mu = \eta_m \mu + (1-\eta_m)\mu_q$
end if

to the unconstrained optimization of $L(\pi)$ w.r.t. mean vector μ and the real valued parameter vector λ such that $\Sigma_{i,i} = \exp(2\max(\lambda_i, \lambda_i - \frac{1}{d}h(\lambda, \beta)))$ with h as defined in Eq. (8).

Proof of Prop. 2 is deferred to App. C. This proposition extends to full covariance matrices Σ where A is its Cholesky decomposition, $\Sigma = AA^T$, by having $A_{i,i} = \exp(\lambda_i)$, real valued off-diagonal entries $\lambda_{\text{off\_diag}}$, and by multiplying A with $\exp(-\frac{1}{d}h(\lambda, \beta))$ if the entropy constraint is violated. This finalizes the projection for constraint (3).

KL constraint. Let us now consider the KL constraint (2) completing constraints of the DPS optimization problem. The KL between two Gaussian distributions $\pi = \mathcal{N}(\mu, \Sigma)$ and $q = \mathcal{N}(\mu_q, \Sigma_q)$ is given by $\mathrm{KL}(\pi \parallel q) = m_q(\mu) + r_q(\Sigma) + e_q(\Sigma)$, where $m_q(\mu) = \frac{1}{2}||\mu - \mu_q||^2_{\Sigma_q^{-1}} = \frac{1}{2}(\mu-\mu_q)^T\Sigma_q^{-1}(\mu-\mu_q)$ is the change in mean, $r_q(\Sigma) = \frac{1}{2}\left(\mathrm{tr}(\Sigma_q^{-1}\Sigma) - d\right)$ is the rotation of the covariance and $e_q(\Sigma) = \frac{1}{2}\log\frac{|\Sigma_q|}{|\Sigma|}$ is the change in entropy.

Proposition 3. *Optimizing any function $L(\pi)$ w.r.t. mean vector μ and covariance Σ of a Gaussian $\pi = \mathcal{N}(\mu, \Sigma)$, under entropy inequality constraint $\mathcal{H}(\pi) \geq \beta$ and KL constraint $\mathrm{KL}(\pi \parallel q) \leq \epsilon$ for Gaussian q such that $\mathcal{H}(q) \geq \beta$ is equivalent to the unconstrained optimization of $L(\pi)$ w.r.t. the parameterization given by Alg. 1.*

Proof sketch. The assumption that $\mathcal{H}(q) \geq \beta$ ensures that the optimization problem admits a valid solution that satisfies both KL and entropy constraint. To prove that the KL constraint is satisfied, we will follow a scheme common to all remaining projections, including projections of discrete distributions. We frame the projection as an interpolation between the input and some target parameters, use bounds stemming from the concavity of the log to obtain simple equations of the interpolation parameter, and solve these equations in closed form. Letting $\mu_{\eta_m} = \eta_m\mu + (1-\eta_m)\mu_q$ and $\Sigma_{\eta_g} = \eta_g\Sigma + (1-\eta_g)\Sigma_q$ be the interpolated mean and covariance for interpolation parameters η_m and η_g in $[0, 1]$, the key result for bounding the KL of $\mathcal{N}(\mu_{\eta_m}, \Sigma_{\eta_g})$ is

$$
\begin{aligned}
|\Sigma_\eta|^{\frac{1}{d}} &\geq |\eta\Sigma|^{\frac{1}{d}} + |(1-\eta)\Sigma_q|^{\frac{1}{d}}, \\
&\qquad \text{(Minkowski determinant inequality)} \\
\log|\Sigma_\eta| &\geq \eta\log|\Sigma| + (1-\eta)\log|\Sigma_q|, \\
&\qquad \text{(concavity of log)} \\
e_q(\Sigma_\eta) &\leq \eta e_q(\Sigma).
\end{aligned}
$$

Exploiting linearity of the trace operator, one can straightforwardly show the same property for $r_q(\Sigma_\eta)$, yielding $r_q(\Sigma_\eta) + e_q(\Sigma_\eta) \leq \eta(r_q(\Sigma) + e_q(\Sigma))$. While for the mean we have $m_q(\mu_\eta) = \eta^2 m_q(\mu)$. From these, one can see by direct computation that the KL and entropy constraints are satisfied with η_m and η_g as in Alg. 1. □

In Prop. 3 we have assumed $\mathcal{H}(q) \geq \beta$ which ensures that q satisfies both entropy and KL constraints and that interpolating a mean and covariance with those of q will reliably return a policy satisfying these constraints. This assumption is rather mild as one would expect entropy to reduce in both DPS and API as data is gathered and hence π shouldn't be constrained to have a higher entropy than q.

The projection defined by Alg. 1 does not necessarily return a search distribution that has KL equal to ϵ if the initial search distribution has KL higher than ϵ. Indeed, when interpolating the covariance matrix one cannot find an interpolation coefficient such that $r_q(\Sigma_\eta) + e_q(\Sigma_\eta)$ is equal to a given target in closed form, as this would require solving $x + \log x = y$ in closed form. The projection we derive in App. A for discrete distributions also relies on concavity of the log to compute an upper bound of the KL. However we show empirically in Sec. 4 for both continuous and discrete distributions that even though the projection is not always on the constraint boundary, optimizing $L \circ g$ will drive the solution to be on the constraint boundary.

3.2. Approximate policy iteration

We extend the previously defined projections to the API case. The policy is now given by $\pi(a|s) \sim \mathcal{N}(\phi(s), \Sigma)$ for state s and arbitrary mean function ϕ, given for example by a neural network. In this setting, the covariance matrix is state-independent and since the entropy only depends on the covariance matrix, the expectation in constraint (7) vanishes. As a result the projection for the entropy constraint is similar to DPS and follows from Prop. 2. For the KL constraint in (6), let $\pi_\eta(.|s)$ be the Gaussian distribution of mean $\eta_m\phi(s) + (1-\eta_m)\phi_q(s)$, with ϕ_q the mean function of previous policy q, and covariance $\eta_g\Sigma + (1-\eta_g)\Sigma_q$. Following the proof of Prop. (3) and by linearity of expectation

$$
\begin{aligned}
\mathbb{E}_s\left[\mathrm{KL}(\pi_\eta(.|s) \parallel q(.|s))\right] \leq \eta_m \mathbb{E}_s\left[m_q^s(\phi(s))\right] \\
+ \eta_g \mathbb{E}_s\left[r_q(\Sigma) + e_q(\Sigma)\right], \quad (9)
\end{aligned}
$$

Algorithm 2 API linear-Gaussian policy projection

Input: A', λ, $\lambda_{\text{off\_diag}}$, $q(.|s) = (A_q^T\psi_q(s), \Sigma_q)$, A^T, ψ, ϵ and β
Output: $\pi(.|s) = \mathcal{N}(A'^T\psi(s), \Sigma)$ complying with KL (6) and entropy (7) constraints
Σ = Entropy_projection($\lambda, \lambda_{\text{off\_diag}}, \beta$)
if $\mathbb{E}_s \text{KL}(\mathcal{N}(A'^T\psi(s), \Sigma) \parallel q(.|s)) > \epsilon$ **then**
  $\eta_g = \frac{\epsilon - m_q(A)}{m_q(A') + r_q(\Sigma) + e_q(\Sigma)}$
  $\Sigma = \eta_g \Sigma + (1 - \eta_g)\Sigma_q$
end if
if $\mathbb{E}_s \text{KL}(\mathcal{N}(A'^T\psi(s), \Sigma) \parallel q(.|s)) > \epsilon$ **then**
  $a = .5\mathbb{E}_s ||A'^T\psi(s) - A^T\psi(s)||^2_{\Sigma_q^{-1}}$
  $b = .5\mathbb{E}_s [(A'^T\psi(s) - A^T\psi(s))^T \Sigma_q^{-1}(A^T\psi(s) - A_q^T\psi_q(s))]$
  $c = m_q(A) + r_q(\Sigma) + e_q(\Sigma) - \epsilon$
  $\eta_m = \frac{-b + \sqrt{b^2 - ac}}{a}$
  $A' = \eta_m A' + (1 - \eta_m)A$
end if

for interpolation parameters η_m and η_g of the mean and covariance respectively. For the mean, m_q^s is given by $m_q^s(\phi(s)) = \frac{1}{2}||\phi(s) - \phi_q(s)||^2_{\Sigma_q^{-1}}$. Since the covariance is state independent, the rightmost expectation in Eq. (9) vanishes. The projection of the KL constraint for state-conditioned Gaussian distributions simply follows by computing η_m and η_g after replacing $m_q(\mu)$ with $\mathbb{E}_s\left[m_q^s(\phi(s))\right]$ in Alg. 1. However the resulting algorithm would be impractical in a reinforcement learning setting. While, in DPS the interpolated mean for a Gaussian distribution can be computed compactly, in API unless ϕ and ϕ_q are linear functions, one would need to store both ϕ and ϕ_q. This scheme would be possible if the policy is built incrementally by adding new components—for example new neurones—at every iteration. In this paper however, we restrict ourselves to training policies of fixed policy class.

To this end we employ a similar scheme to the two time-scale RL algorithms (Levine et al., 2017; Chung et al., 2019), and split the mean function $\phi(s) = A^T\psi(s)$ into the feature part ψ and the linear part A. We then use standard API algorithms to update the policy including the feature part before using the optimization tools we develop in this section to further optimize the linear part and covariance matrix of the distribution. We experiment this scheme with two API algorithms discussed in Sec. 4.1. Starting from the data generating policy q of mean function $\phi_q = A_q^T\psi_q(s)$, API is used to update q and obtain an intermediary policy $\phi = A^T\psi$. The only assumption on the API algorithm is that the intermediate policy does not violate the constraints—which can be enforced by e.g. being overly conservative with the step-size and backtracking if necessary. For the following proposition let $m_q(A) = \mathbb{E}_s\left[m_q^s(A^T\psi(s))\right]$.

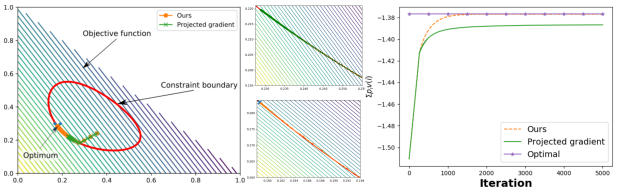

Figure 1. Depiction of a linear optimization problem in probability space under an entropy constraint, and comparison with the projected gradient method (right). See Sec. 4 for details.

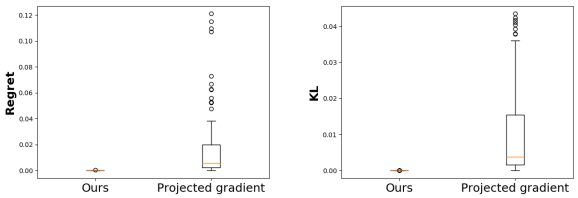

Figure 2. Regret and KL to the optimal solution of our method and the projected gradient method, on 100 random instances of the problem described in Sec. 4.

Proposition 4. *Optimizing any function $L(\pi)$ w.r.t. parameters A' and Σ of linear in feature Gaussian policy $\pi(.|s) = \mathcal{N}(A'^T\psi(s), \Sigma)$, under entropy constraint (7) and KL constraint (6) to linear in feature Gaussian policy $q(.|s) = \mathcal{N}(A_q^T\psi_q(s), \Sigma_q)$ such that i) $\mathcal{H}(q) \geq \beta$ and ii) there exist A such that $m_q(A) \leq \epsilon$, is equivalent to the unconstrained optimization of $L(\pi)$ w.r.t. the parameterization given by Alg. 2.*

Assumption (ii) on q ensures that the feature change from ψ_q to ψ does not preclude the existence of a solution to the optimization problem. Proof of Prop. 4 is deferred to App. C. To summarize our contributions, two RL formulations with constrained updates were considered. To solve the constrained problem we proposed parameterizations and associated projections to transform the update problems to unconstrained ones. The biggest advantage of these projections is that they are differentiable—at least outside of the constraint boundary. The practical interest of this property will be made apparent in the experiments section. Implementation of Alg. 2 is provided in https://github.com/akrouriad/papi.

4. Experiments

Our first set of experiments is on simple optimization problems to assess the validity of our proposed optimization scheme for constrained problems. Most of the introduced projections g are not on the constraint boundary, at the exception of the entropy constraint of a Gaussian distribution. Thus, it remains to be seen if optimizing $L \circ g$ by gradient ascent can match the quality of solutions obtained via the

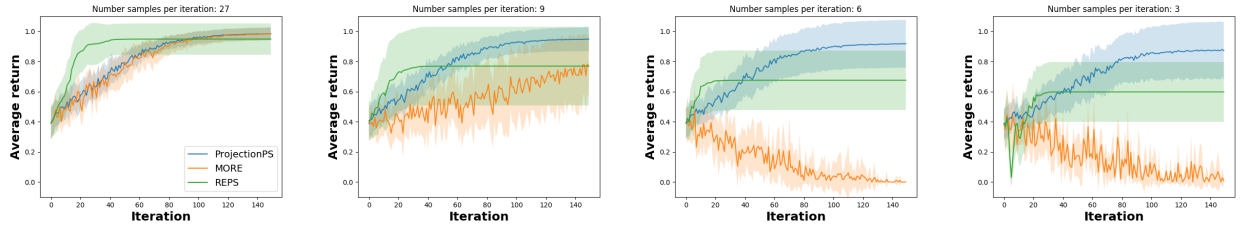

Figure 3. DPS performance on smooth objective functions. Left to right, samples per iteration of 27, 9, 6, and 3. Averaged over 11 runs.

method of Lagrange multipliers on simple problems.

We first consider the linear problem in probability space of maximizing $L(p) = \sum p_i v_i$ over a discrete distribution p under entropy constraint $\mathcal{H}(p) \geq \beta$, for some arbitrary vector v. Following the method of Lagrange multipliers, the optimal solution has shape $p_i \propto \exp(v_i/\eta)$ where η is a dual parameter that can be accurately computed by optimizing a uni-dimensional, convex, dual function. Using our method we optimize the unnormalized log probabilities l and use the projection g in App. A to ensure that $\mathcal{H}(g(l)) \geq \beta$ for any parameter l. Whereas we optimize $L \circ g$, we compare our method to the Projected Gradient method (Bertsekas, 1999) that projects back to the acceptable region *after* each gradient step. We use for that the same projection g.

Fig. 1 shows in probability space an example run of such an experiment for a three dimensional parameter l. While the projection g is not on the constraint boundary, gradient of $L \circ g$ drives the optimizer to the optimal point. Whereas using the same projection, the method of projected gradient stops at a point where it bounces in and out of the valid region without moving towards the optimum. This experiment was repeated over 100 independent runs with randomly sampled v and an initial point in the valid region. Fig. 2 shows that our method always finds solutions with very small regret and KL divergence to the optimal solution. While the average regret and KL for the projected gradient method is orders of magnitude higher. Despite g not projecting on the constraint boundary, optimizing $L \circ g$ finds near-optimal solutions. In contrast, using the same projection but projecting after gradient updates does not yield as good results. However, when the initial point is not in the valid region there were few cases where our method would converge to a stationary point that is not optimal; indicating that $L \circ g$ is not a convex function despite the optimization problem being convex.

Secondly, we evaluate the use of the projection in Alg. 1 for the optimization of randomly generated smooth functions (mixture of 25 bivariate Gaussians). Our approach is compared to two baselines, REPS (Peters et al., 2010) and MORE (Abdolmaleki et al., 2015) that solve a similar problem to the one introduced in Sec. 3.1. For this problem the method of Lagrange multipliers provides a closed form solution but not when the search distribution is constrained to be Gaussian. REPS and MORE use sample based approximations to exploit the closed form solution for arbitrary distributions. We consider a more direct approach, termed 'ProjectionPS', that optimizes $L \circ g$ where g is given by Alg. 1. Fig. 3 shows that all three methods perform similarly when the sample count per iteration is high but that our direct approach is more robust to low sample counts. Additional figures in App. B show that despite g not projecting on the constraint boundary of the KL constraint, the returned distributions have almost always a KL close to the constraint limit ϵ. Additional details for this experiment are provided in App. B.

4.1. API continuous action benchmarks

The use of the projections proposed in Sec. 3 in improving existing API algorithms is now assessed. We combine the projection of Alg. 2 to both TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017). TRPO solves the same API problem defined in Sec. 2.2 except for the entropy constraint (7) and is thus a natural baseline to compare to. The tools we have introduced can be integrated to several other API algorithms. PPO was chosen as the second base for our projections because Prop. 4 requires the features to only slowly change. While PPO does not impose a hard constraint on the change in distribution between successive iterations, the clipped loss still proved useful in controlling the latter. Indeed, the clipped loss of PPO discourages $\left|\frac{\pi(a|s)}{q(a|s)} - 1\right|$ to be too large. This quantity when taken in expectation of q, as is the case in PPO, is no other than $\mathbb{E}_{a\sim q}\left|\frac{\pi(a|s)}{q(a|s)} - 1\right| = \int |\pi(a|s) - q(a|s)| da$, two times the total variation between $\pi(.|s)$ and $q(.|s)$. Hence, the clipped loss of PPO provides no incentive in changing the policy past a certain total variation threshold. By adjusting the step-size of the gradient descent algorithm we were able to reliably update the non-linear part of the policy such that the KL constraint is respected while relevant features are learned.

We implement our projections within OpenAI's code base (Dhariwal et al., 2017). The reported results for the baselines are also obtained from this implementation. The only

Table 1. Average trajectory reward of the initial 100 iterations (Init.) and average reward of best window of 500 trajectories (Best) averaged over 11 runs. Bolded are significantly better according to Welch's t-test with p-value $< .05$.

	RSHopper-v1		RSWalker-v1		RSHalfCheetah-v1		RSAnt-v1	
	Init.	Best	Init.	Best	Init.	Best	Init.	Best
PAPI-PPO	**121 ± 8**	**2237 ± 56**	**75 ± 4**	**1426 ± 434**	**55 ± 4**	2353 ± 67	**497 ± 29**	**1924 ± 110**
PAPI-TRPO	75 ± 7	**2166 ± 204**	60 ± 5	**1712 ± 382**	32 ± 4	**2524 ± 113**	456 ± 29	1684 ± 233
PAPI-0-TRPO	61 ± 5	**2122 ± 225**	51 ± 3	**1633 ± 342**	20 ± 2	2202 ± 158	415 ± 17	1610 ± 201
TRPO	87 ± 11	**2180 ± 123**	65 ± 3	1183 ± 372	40 ± 2	1882 ± 172	443 ± 28	1575 ± 220
PPO	68 ± 8	2024 ± 175	58 ± 1	1016 ± 359	30 ± 2	2265 ± 85	420 ± 18	**1871 ± 86**

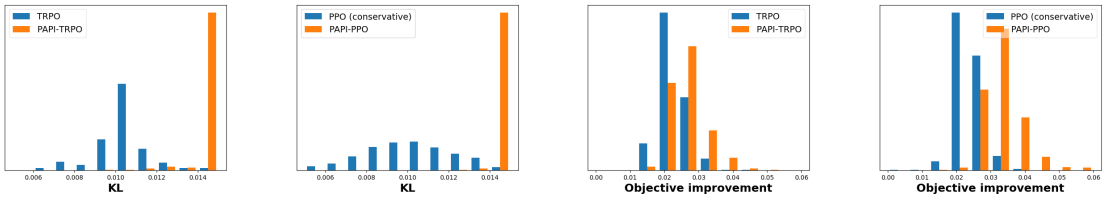

Figure 4. Effect of additional optimization steps using our optimization approach on solutions returned by TRPO and a conservative version of PPO. Results show that our method finds policies closer to the KL constraint with higher objective improvement.

modification we make to the implementation of TRPO and PPO is to use a Gaussian policy with a full covariance matrix to match our setting. All our experiments use a neural network policy with two hidden layers of 64 neurones. We implement the optimization of the last layer and the covariance matrix over a conservative version of PPO obtained by saving the policy after each epoch and upon completion, backtracking to the last set of parameters that has a KL less than ϵ. If the selected policy is not part of the last 4 epochs the step-size of PPO's optimizer is reduced. We then proceed by doing several optimization steps using Alg. 2 to optimize the last layer of the neural network, i.e. the linear part, and the covariance matrix of the Gaussian policy. TRPO on the other hand did not require special modifications and our optimization was performed after the standard TRPO update which always ensures that the KL is less than ϵ. In the following, each of PPO and TRPO are prefixed with PAPI (for Projected API) to indicate that additional optimization steps were performed using the proposed tools.

We first study the behavior of each algorithm in solving the constrained policy update of Sec. 2.2. For this we run each of the base algorithms on an RL task (here RoboschoolHopper) and record the average KL and the improvement in the update objective $L(\pi) - L(q)$. We then optimize $L \circ g$ using the projection of Alg. 2 and record the new KL and objective improvement on the same update data set. Fig. 4 shows a clear improvement of PAPI-TRPO and PAPI-PPO over their base algorithm in optimizing the policy update objective. Notably, PAPI-TRPO finds solutions closer to the KL constraint boundary and with higher objective value than those obtained by the quadratic approximation of the KL constraint used by TRPO.

In DPS, we noticed that performing additional gradient descent steps during search distribution update would only help the overall performance of the algorithm. We did not observe the same relation in API. It was already known that an unconstrained maximization of L could lead to poor performance. However, even by constraining the KL between successive policies, obtaining a better solution to the policy update problem as defined in Sec. 2.2 does not necessarily translates to better end performance. We observed that an important indicator to track was the matrix norm of A, the linear part of the policy mean. If the norm increases too fast, it would lead to premature convergence of the algorithm.

A remedy was to use mini-batches when optimizing $L \circ g$ which had a regularizing effect on A. For all of the experiments—including Fig. 4—PAPI-PPO refers to performing 20 epochs with mini-batches of size 64. For the entropy constraint, we adopt a two phase approach where we initially do not constrain the entropy until it reaches half of the initial entropy and then decrease β linearly by a fixed amount of ϵ. Using the same parameters for PAPI-TRPO would result in improvements over TRPO for some tasks but the entropy of the final policy was always relatively high. We obtained best performance for PAPI-TRPO by enforcing an entropy equality constraint using Prop. 1 and only optimizing A for 10 epochs with mini-batches of size 64. To isolate the impact of the entropy equality constraint we additionally report performance of PAPI-0-TRPO which does not do any additional optimization of the policy but performs the TRPO update under entropy constraint using Prop. 1. As for baselines, from here on PPO refers to the default version of the algorithm and not the conservative version which is only used for PAPI-PPO.

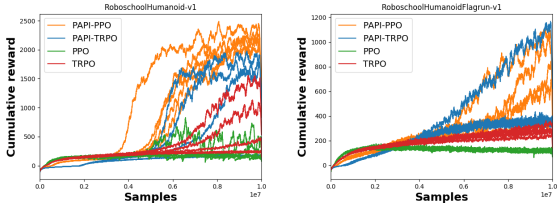

Figure 5. Comparison between PAPI-PPO and PAPI-TRPO with their base counterpart on complex RL tasks.

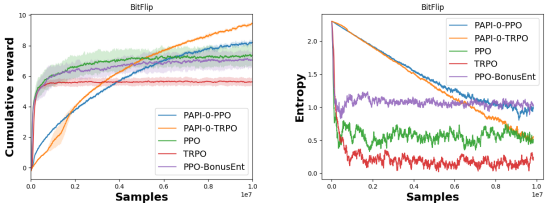

Figure 6. Policy return on the BitFlip task averaged over 11 runs (left). Policy entropy of the policy in the first run (right).

We run a first set of experiments on four benchmark tasks from Roboschool[3] (Brockman et al., 2016). Tab. 1 reports two metrics for each task. The average trajectory reward of the initial 100 iterations (equivalent to 320K samples) to measure learning speed and average reward of best window of 500 trajectories to measure peak performance of each algorithm. Results are averaged over 11 runs. We observe that the additional optimization steps of $L \circ g$ provide a clear performance boost in the initial learning iterations both from the performance of PAPI-PPO and from the improvements of PAPI-TRPO over PAPI-0-TRPO which does not perform any additional optimization step. As for peak performance we did not observe big discrepancies between the base versions and the PAPI versions except for the Walker (PPO and TRPO) and HalfCheetah (TRPO only). In both cases we hypothesize that the biggest improvement in peak performance is the result of the entropy constraint, judging by the performance of PAPI-0-TRPO. More challenging problems are tackled in the next section.

4.2. Problems with hard exploration

We extend the study of the previous section to more challenging tasks of the Roboschool testbed. For these tasks we let the RL agents collect up to 10 million samples and launch 5 independent runs for each algorithm. All other hyperparameters are kept as in the previous section. Fig 5 shows a clear improvement of both PAPI-TRPO and PAPI-PPO over their base algorithm with special mention to PAPI-PPO that consistently learns good policies on both tasks.

Finally, we tackle a discrete action task designed to require sustained exploration, termed BitFlip. In BitFlip, the state space is a vector of N bits and there are N actions, flipping the value of each bit. The reward is given by $r(s, a, s') = -\text{val}(s')$ if a flips a bit to 1 and $\text{val}(s)$ otherwise, where $\text{val}(s)$ is the numerical value of the bit vector s. All bits of s_0 are 0 and the optimal policy is to continuously flip the bits from right to left. This problem is challenging because the optimal policy has to choose roughly half of the time the action that does not provide the highest immediate reward.

[3]The tasks are not directly comparable to their analogue in Mujoco (Erez et al., 2015). Please see for example Srouji et al. 2018 for a comparison between the two testbeds.

We compare TRPO and PPO with variants having a strict entropy constraint, using the projection defined in App. A, and linearly decreasing constraint lower bound β. A variant of PPO with an entropy bonus (Schulman et al., 2017), instead of a hard constraint, is also included. Fig. 6 shows that TRPO converges early on to a sub-optimal policy. The entropy of PPO and PPO-BonusEnt plateaus at a higher level than TRPO which allows both baselines to steadily improve, but because of the abrupt initial entropy reduction the improvement is very slow. The entropy bonus, while having a clear effect on the entropy of the policy is not as transparent as the imposed hard constraint. Because the balancing between L and the entropy bonus is fixed, it initially has no effect on the policy entropy before reaching a point where it dominates the objective. In contrast, the effect of the linear decrease of β is clear in Fig. 6 on the entropy plot. As for its impact on the return, it slows PAPI-0-TRPO and PAPI-0-PPO initially but sustains a fast increase of the policy return, resulting in a significantly better final policy. Exploration in RL is complex, and the proposed projections to control entropy of both continuous and discrete policies can only be see as heuristics. However, we have shown that they have a clear impact on the end performance while being easier to tune than adding an entropy bonus.

5. Conclusion

We introduced a set of projections to tackle constrained optimization problems common in RL. These projections were empirically evaluated on a variety of settings from simple optimization problems to complex RL tasks. The practical interest of the adopted optimization scheme has been demonstrated against many baselines such as projected gradient descent and several DPS and API algorithms. The proposed projections can be integrated to virtually any RL algorithm. While adding a KL constraint might require modification of the RL algorithm, projections for the entropy constraint are comparatively lightweight. They are best thought of as just adding a layer—the projection—to the policy, while potentially reaping all benefits discussed in the experiments section. We hope that such a scheme for controlling exploration of the policy will prove to be a valid alternative to adding an entropy bonus to the reward.

Acknowledgements

The research leading to these results has received funding from NVIDIA, from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 640554 (SKILLS4ROBOTS) and from DFG project PA 3179/1-1 (ROBOLEAP). Computations were conducted on the Lichtenberg high performance computer of TU Darmstadt and the NVIDIA DGX station.

References

Abdolmaleki, A., Lioutikov, R., Peters, J., Lau, N., Pualo Reis, L., and Neumann, G. Model-based relative entropy stochastic search. In *Advances in Neural Information Processing Systems (NIPS)*. 2015.

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International Conference on Machine Learning (ICML)*, 2017.

Akrour, R., Sorokin, D., Peters, J., and Neumann, G. Local bayesian optimization of motor skills. In *International Conference on Machine Learning (ICML)*, 2017.

Akrour, R., Abdolmaleki, A., Abdulsamad, H., Peters, J., and Neumann, G. Model-free trajectory-based policy optimization with monotonic improvement. *Journal of Machine Learning Resource (JMLR)*, 2018.

Arenz, O., Zhong, M., and Neumann, G. Efficient gradient-free variational inference using policy search. In *International Conference on Machine Learning (ICML)*, 2018.

Bagnell, J. A. and Schneider, J. C. Covariant Policy Search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

Bertsekas, D. P. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

Bertsekas, D. P. Approximate policy iteration: a survey and some new methods. *Journal of Control Theory and Applications*, 9(3):310–335, Aug 2011.

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., and Lee, M. Natural actor-critic algorithms. *Automatica*, 2009.

Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Chung, W., Nath, S., Joseph, A., and White, M. Two-timescale networks for nonlinear value function approximation. In *International Conference on Learning Representations*, 2019.

Conti, E., Madhavan, V., Petroski Such, F., Lehman, J., Stanley, K., and Clune, J. Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Advances in Neural Information Processing Systems (NIPS)*. 2018.

da Silva, B., Konidaris, G., and Barto, A. Learning Parameterized Skills. In *International Conference on Machine Learning (ICML)*, 2012.

Deisenroth, M. P., Neumann, G., and Peters, J. A Survey on Policy Search for Robotics. *Foundations and Trends in Robotics*, pp. 388–403, 2013.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. Openai baselines. https://github.com/openai/baselines, 2017.

Erez, T., Tassa, Y., and Todorov, E. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ODE and physx. In *International Conference on Robotics and Automation (ICRA)*, 2015.

Heidrich-Meisner, V. and Igel, C. Hoeffding and bernstein races for selecting policies in evolutionary direct policy search. In *International Conference on Machine Learning (ICML)*, 2009.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 267–274, 2002.

Levine, N., Zahavy, T., Mankowitz, D. J., Tamar, A., and Mannor, S. Shallow updates for deep reinforcement learning. In *Neural Information Processing Systems (NIPS)*, 2017.

Levine, S. and Abbeel, P. Learning Neural Network Policies with Guided Policy Search under Unknown Dynamics. *Advances in Neural Information Processing Systems*, pp. 1–3, 2014a.

Levine, S. and Abbeel, P. Learning neural network policies with guided policy search under unknown dynamics. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 1071–1079. Curran Associates, Inc., 2014b.

Parisi, S., Abdulsamad, H., Paraschos, A., Daniel, C., and Peters, J. Reinforcement learning vs human programming in tetherball robot games. In *International Conference on Intelligent Robots and Systems (IROS)*, 2015.

Peters, J. and Schaal, S. Natural Actor-Critic. *Neurocomputation*, 71(7-9):1180–1190, 2008. ISSN 0925-2312.

Peters, J., Mülling, K., and Altün, Y. Relative entropy policy search. In *National Conference on Artificial Intelligence (AAAI)*, 2010.

Pirotta, M., Restelli, M., Pecorino, A., and Calandriello, D. Safe policy iteration. In Dasgupta, S. and McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pp. 307–315. JMLR Workshop and Conference Proceedings, May 2013.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. Parameter space noise for exploration. *CoRR*, 2017.

Scherrer, B. Approximate policy iteration schemes: A comparison. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 1314–1322, 2014.

Schulman, J., Levine, S., Jordan, M., and Abbeel, P. Trust Region Policy Optimization. *International Conference on Machine Learning (ICML)*, pp. 16, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Srouji, M., Zhang, J., and Salakhutdinov, R. Structured control nets for deep reinforcement learning, 2018.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, Boston, MA, 1998.

Szepesvari, C. *Algorithms for Reinforcement Learning*. Morgan & Claypool, 2010.

Tangkaratt, V., Abdolmaleki, A., and Sugiyama, M. Guide actor-critic for continuous control. In *International Conference on Learning Representations*, 2018.

Tassa, Y., Mansard, N., and Todorov, E. Control-limited differential dynamic programming. In *International Conference on Robotics and Automation (ICRA)*, 2014.

Thomas, P. S. and Okal, B. A notation for markov decision processes, 2015.

Todorov, E. and L., W. A generalized Iterative LQG Method for Locally-Optimal Feedback Control of Constrained Nonlinear Stochastic Systems. In *American Control Conference (ACC)*, 2005.