Value constrained model-free continuous control

Steven Bohez, Abbas Abdolmaleki, Michael Neunert,
Jonas Buchli, Nicolas Heess, Raia Hadsell
DeepMind
London, UK

Abstract

Applying Reinforcement Learning algorithms to continuous control problems often results in policies which rely on high-amplitude, high-frequency control signals, known colloquially as *bang-bang* control. To counteract this issue, multi-objective optimization can be used to simultaneously optimize both the reward and some auxiliary cost that discourages undesired control. In this paper we propose a new constraint-based approach which defines a lower bound on the return while minimizing one or more costs (such as control effort). We employ Lagrangian relaxation to learn both (a) the parameters of a control policy that satisfies the desired constraints and (b) the Lagrangian multipliers for the optimization. Moreover, we learn a single conditional policy that is able to dynamically change the trade-off between return and cost.[1]

1 Introduction

Deep Reinforcement Learning (RL) has achieved numerous successes over the last couple of years, enabling learning of effective policies from high-dimensional input, such as pixels, on complicated tasks. However, compared to problems with discrete action spaces, control problems with high-dimensional continuous state-action spaces – as often encountered in robotics – have proven much more challenging. Beyond the issue of exploration in high-dimensional continuous action spaces, RL algorithms rarely learn policies that produce smooth control signals when just optimizing for task success. Instead, the control signals have a tendency to switch between extreme values at high-frequency, a phenomenon colloquially referred to as *bang-bang* control. Smoothness, however, is a desirable property in most real-world control problems. Unnecessary oscillations are not only energy inefficient, they also exert stress on a physical system by exciting second-order dynamics and increasing wear and tear on structural elements and actuators.

To regularize the behavior, one can add penalties to the reward function. As a result, the reward function is composed of positive reward for achieving the goal and negative reward (penalties) for control action discontinuities or high energy use. This effectively casts the problem into a multi-objective optimization setting, where – depending on the ratio between the reward and the different penalties – different behaviors may be achieved. While every ratio will have its optimal policy, finding the ratio that results in the desired behavior, i.e. smooth control while still achieving an acceptable task success rate, can be difficult and can require extensive hyperparameter tuning. Often, one must find different hyperparameter settings for different reward-penalty trade-offs or tasks. The process of finding appropriate parameter values can be tedious and cumbersome, and may prevent robust general solutions. In this paper we rephrase the problem: instead of trying to find the right ratios between reward and penalties, we regularize the optimization problem by adding constraints, thereby reducing its effective dimensionality. More specifically, we propose to minimize the penalty with respect to a lower bound on the success rate of the task.

[1] Videos available at `https://sites.google.com/view/minitaurphys2018`

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

2 Constrained optimization for control

In the classical Markov Decision Process (MDP) setting, an agent sequentially interacts with an environment by observing the state of the environment $\boldsymbol{s}$ and taking an action according to a policy $\boldsymbol{a} \sim \pi(\boldsymbol{s} \mid \boldsymbol{s})$. Each action causes a state transition with an associated reward defined by some function $r(\boldsymbol{s}, \boldsymbol{a})$. The goal of the agent is to maximize the return, $\max_\pi \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\pi}\left[\sum_t r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right]$. In the Constrained Markov Decision Process (CMDP) [3] setting we add cost function $c(\boldsymbol{s}, \boldsymbol{a})$. The goal is to find a policy $\pi(\boldsymbol{a} \mid \boldsymbol{s})$ that trades-off between maximizing the (expected) reward and minimizing the cost by imposing hard constraints to reduce dimensionality. For example, in locomotion, desired behavior can be defined in terms of a lower bound on speed or an upper bound on an energy cost. While a constraint can be placed on either the reward or the cost, in this work we consider a lower bound on the total return. In practice one often optimizes the γ-discounted return in both cases. We define the action-value function as $Q_r(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\pi}\left[\sum_t \gamma^t \cdot r(\boldsymbol{s}_t, \boldsymbol{a}_t) \,|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}\right]$, construct in a similar fashion the expected discounted cost action-value function $Q_c(\boldsymbol{s}, \boldsymbol{a})$.. This allows us to define the CMDP as follows

$$\min_\pi \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\pi}\left[Q_c(\boldsymbol{s}, \boldsymbol{a})\right], \text{s.t. } \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\pi}\left[Q_r(\boldsymbol{s}, \boldsymbol{a})\right] \geq V_r^*. \tag{1}$$

Generally the constraint in Equation 1 is not satisfied at the start of learning, as the agent first needs to learn how to solve the task. This limits the choice of methods that can be used to solve the CMDP: Many existing methods assume that the constraint is satisfied at the start and only ensure that the solution remains within the constraint-satisfying regime [e.g. 2]. Lagrangian relaxation is a general method for solving general constrained optimization problems including CMDPs [3]. In this setting, the hard constraint is relaxed into a soft constraint, where any constraint violation acts as a penalty for the optimization. Applying Lagrangian relaxation to Equation 1 results in the unconstrained dual problem

$$\max_\pi \min_{\lambda\geq 0} \mathbb{E}_{\mathbf{s},\mathbf{a}\sim\pi}\left[Q_\lambda(\boldsymbol{s}, \boldsymbol{a})\right], \text{with } Q_\lambda(\boldsymbol{s}, \boldsymbol{a}) = \lambda\left(Q_r(\boldsymbol{s}, \boldsymbol{a}) - V_r^*\right) - Q_c(\boldsymbol{s}, \boldsymbol{a}), \tag{2}$$

with an additional minimization objective over the Lagrangian multiplier λ. A larger λ results in a higher penalty for violating the constraint. Hence, we can iteratively update λ by gradient descent on $Q_\lambda(\boldsymbol{s}, \boldsymbol{a})$ until the constraint is satisfied. Under assumptions described in Tessler et al. [6], this approach converges to a saddle point. To perform the outer policy optimization for π any off-the-shelf off-policy optimization algorithm can be used. In this work, we use MPO [1]. To improve the stability of training, we also perform a change of variable $\lambda = \exp(\lambda')$, $\lambda' \in \mathbb{R}$, and normalize $Q_\lambda(\boldsymbol{s}, \boldsymbol{a})$ by $\exp(\lambda') + 1$. One downside of the CMDP formulation given in Equation 1 is that the constraint is placed on the *expected* value. This implies that the constraint will not necessarily be satisfied at every single timestep, or visited state, during the episode. We can extend the single constraint introduced previously to a set, possibly infinite, of point-wise constraints; one for each state induced by the policy: $\forall \mathbf{s} \sim \pi : \mathbb{E}_{\mathbf{a}\sim\pi}\left[Q_r(\boldsymbol{s}, \boldsymbol{a})\right] \geq V_r^*$. As before, this problem can be optimized with Lagrangian relaxation by introducing state-dependent Lagrangian multipliers:

$$\max_\pi \mathbb{E}_{\mathbf{s}\sim\pi}\left[\min_{\lambda(\boldsymbol{s})\geq 0} \mathbb{E}_{\mathbf{a}\sim\pi}\left[Q_\lambda(\boldsymbol{s}, \boldsymbol{a})\right]\right], \text{with } Q_\lambda(\boldsymbol{s}, \boldsymbol{a}) = \lambda(\boldsymbol{s})\left(Q_r(\boldsymbol{s}, \boldsymbol{a}) - V_r^*\right) - Q_c(\boldsymbol{s}, \boldsymbol{a}). \tag{3}$$

Here we have made the assumption that nearby states have similar λ multipliers and we can generalize to unseen states, analogues to value estimates. In practice, we train a single critic model that outputs $\lambda(\boldsymbol{s})$ as well as $Q_c(\boldsymbol{s}, \boldsymbol{a})$ and $Q_r(\boldsymbol{s}, \boldsymbol{a})$. In general a point-wise constraint with fixed lower bound might be impossible to satisfy for some states in a given task (e.g. we cannot satisfy a reward constraint in the swing-up phase of the simple pendulum). However, the lower bound can also be made state-dependent and our approach will still be applicable.

Up to this point, we have made the assumption that we are only interested in a single, fixed value for the lower bound. However, in some tasks one would want to solve Equation 3 for different lower bounds V_r^*, i.e. minimizing cost for various success rates. For example, in a locomotion task, one could be interested in optimizing energy for multiple different target speeds or gaits. To avoid the need to solve a large number of optimization problems, we can condition the policy, value function and Lagrangian multipliers on the desired target value and learn a bound-conditioned policy. Such a conditional constraint allows a single policy to dynamically trade off cost and return.

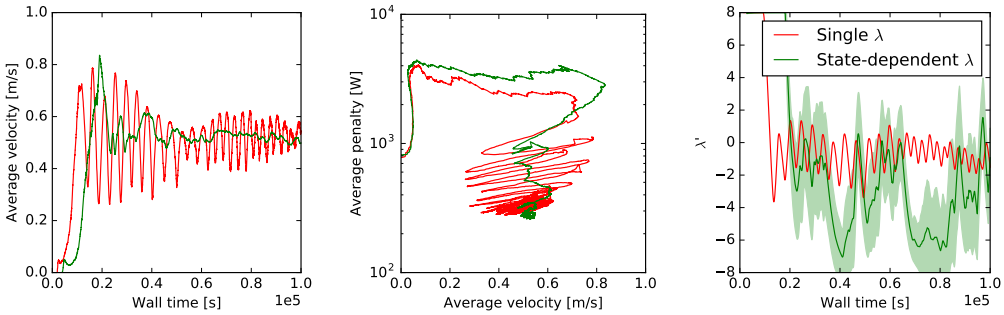

Figure 1: Comparison of a single versus a state-dependent λ multiplier for models trained to achieve a minimum velocity of 0.5m/s.

3 Experiments

We apply our approach to a realistic, energy-optimized robotic locomotion task, using the Minitaur quadruped developed by Ghost Robotics [4]. Learning-based approaches have shown promise as a way to devise locomotion controllers that show versatile and very dynamic behaviors [5]. We model the Minitaur in MuJoCo [7] using model parameters obtained from data sheets as well as system identification. The observations of the RL agent include noisy motor positions, yaw, pitch, roll, angular velocities and accelerometer readings. The policy generates setpoints for low-level position controller with a timestep of 10ms. As we are only considering forward locomotion, we set the reward $r(\boldsymbol{s}, \boldsymbol{a})$ to be the forward velocity of the robot. We do however want learned gaits to be sufficiently well-behaved. One way to achieve smooth control and locomotion is to optimize for energy efficiency, as fast, opposing actions typically require more power. We hence set the cost $c(\boldsymbol{s}, \boldsymbol{a})$ is set to be the total power usage of the motors. For both the policy and the critic we use a two-layer MLP with 300 and 200 ReLU units, followed by an LSTM of 100 cells to cope with the noisy partial observations.

Figure 1 shows a comparison between learning dynamics between a model using a single λ multiplier and a model with a state-dependent one. Both agents try to achieve a lower bound on the value that is equivalent to a minimum velocity of 0.5m/s. At first, both agents "focus" on satisfying the constraint, increasing the penalty significantly in order to do so. Once the target velocity is exceeded, the agents start to optimize the penalty, which drives them back to the imposed bound. A single multiplier that is applied to all states leads to larger changes in behavior space, where the agent oscillates between moving too slow at a lower penalty or too fast at a higher penalty. The agent with the state-dependent multiplier tracks the target velocity more closely, and achieves slightly lower penalties. Looking at the λ values over time in Figure 1, we see that they are generally lower in the latter case as well.

In Table 1, we compare the reward-penalty trade-off for settings with a fixed lower bound on the velocity. We compare our approach to baselines where we clip the reward as $r'(\boldsymbol{s}_t, \boldsymbol{a}_t) = \min(r(\boldsymbol{s}_t, \boldsymbol{a}_t), r^*)$ and use a fixed coefficient α. As there is less incentive for the agent to increase the reward over r^*, there is more opportunity to optimize the penalty. Results shown are the per-step error with respect to the desired target velocity and the penalty, averaged across 4 seeds and 100 episodes each. We observe that when α is set too high, the agent is biased towards standing still.

Table 1: Results for models trained to achieve a fixed lower bound on the velocity. Reported numbers are average per-step (velocity error [m/s], penalty [W]). We highlight the best constant α, in terms of error, for each target bound.

Target	$\alpha = 3\text{e}{-3}$		$\alpha = 1\text{e}{-3}$		$\alpha = 3\text{e}{-4}$		$\alpha = 1\text{e}{-4}$		**Constraint**	
	error	penalty	error	penalty	error	penalty	error	penalty	error	penalty
0.1	-0.1,	35.74	**-0.01**,	**104.2**	0.07,	112.35	0.1,	245.49	**0.01**,	**127.14**
0.2	-0.2,	46.48	**-0.01**,	**210.04**	0.15,	207.19	0.23,	399.83	**0.03**,	**106.88**
0.3	-0.3,	50.3	**0.06**,	**154.91**	0.16,	213.1	0.24,	429.6	**0.04**,	**89.97**
0.4	-0.4,	54.05	**0.06**,	**195.98**	0.11,	306.1	0.32,	627.66	**0.05**,	**132.97**
0.5	-0.5,	60.71	**0.13**,	**250.69**	**0.13**,	**332.53**	0.26,	808.38	**0.05**,	**142.93**

Table 2: Results of models that are conditioned on the target velocity, evaluated for for different values.

Target	$\alpha = 3\text{e}{-3}$		$\alpha = 1\text{e}{-3}$		$\alpha = 3\text{e}{-4}$		$\alpha = 1\text{e}{-4}$		**Constraint**	
	error	penalty	error	penalty	error	penalty	error	penalty	error	penalty
0.0	**0.0**,	**53.68**	0.01,	116.59	0.17,	272.45	0.37,	757.53	**0.0**,	**84.07**
0.1	-0.1,	54.49	**0.0**,	**158.68**	0.21,	324.16	0.37,	619.3	**0.0**,	**141.86**
0.2	-0.2,	53.54	**0.02**,	**256.68**	0.21,	373.13	0.36,	627.19	**0.04**,	**174.79**
0.3	-0.3,	53.6	**-0.02**,	**314.71**	0.16,	336.48	0.42,	747.24	**0.02**,	**188.18**
0.4	-0.4,	54.82	**-0.07**,	**384.94**	0.15,	467.21	0.32,	870.34	**0.05**,	**252.54**
0.5	-0.5,	52.37	-0.1,	366.48	**0.01**,	**594.36**	0.27,	1026.3	**0.05**,	**361.16**

In all other cases, the targeted speed is exceeded by some margin that increases with decreasing α. While there is less incentive to exceed r^*, a larger margin decreases the chances of the actual speed momentarily dropping below the target speed. Using the constraint-based approach, we generally achieve average actual speeds closer to the target speed and at a lower average penalty.

Table 2 shows a comparison between agents are trained across varying target speeds, observed by the agent, sampled uniformly in $[0, 0.5]$ m/s. In this case, the same conditional policy is evaluated for multiple target values. We make similar observations: a fixed penalty coefficient generally leads to higher speeds then the set target, and higher penalties. Interestingly, for higher target velocities, the actual velocity exceeds the target less, indicating that different values for α are required for different targets. As we learn multipliers that are conditioned on the target, we can track the target more closely, even for higher speeds, while simultaneously achieving a lower penalty.

Videos showing some of the learned behaviors, both in the fixed and conditional constraint case, can be found at `https://sites.google.com/view/minitaurphys2018`.

4 Conclusion

In order to regularize behavior in continuous control RL tasks, we introduced a constraint-based approach that is able to automatically trade off rewards and penalties. Specifically, we minimize the penalties with respect to a point-wise lower bound on the reward value, for each state that the learned policy encounters. The resulting constrained optimization problem is solved using Lagrangian relaxation and learning the resulting state-dependent Lagrangian multipliers, allowing generalization of multipliers across states. The policy and critic can furthermore generalize across lower bounds by making the constraint value observable, resulting in a single bound-conditional RL agent that is able to dynamically trade off reward and costs in a controllable way. In a simulated locomotion task with the Minitaur quadruped, we are able to minimize electrical power usage with respect to a lower bound on the forward velocity. We also learn a single, goal-conditioned policy that is able to move efficiently across a range of target velocities.

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a Posteriori Policy Optimisation. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=S1ANxQW0b`.

[2] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning*, pages 22–31, 2017.

[3] E. Altman. *Constrained Markov Decision Processes*. Chapman and Hall, 1999.

[4] G. Kenneally, A. De, and D. E. Koditschek. Design principles for a family of direct-drive legged robots. *IEEE Robotics and Automation Letters*, 1(2):900–907, July 2016.

[5] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *Robotics: Science and Systems (RSS)*, 2018.

[6] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. Reward constrained policy optimization. *CoRR*, abs/1805.11074, 2018. URL `https://arxiv.org/abs/1805.11074`.

[7] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, Oct 2012. doi: 10.1109/IROS.2012.6386109.