Mirror Descent for Robust Reinforcement Learning

Shivam Garg SHIVAM.GARGCD.CSE14@ITBHU.AC.IN
Indian Institute of Technology (BHU), Varanasi – 221005 (UP) India.

Abstract

Robust Reinforcement Learning deals with problems where the transition probabilities of the environment are uncertain or only an approximate simulation of the environment is available to the agent. This uncertainty can be modeled by assuming the probabilities to be composed of a true distribution plus some confidence region. Using this formulation, we give two new gradient based algorithms for function approximators which use the Mirror Descent Optimization technique.

1. Introduction

Reinforcement Learning (RL) refers to the broad range of algorithms which learn goal directed optimum decision making. In RL problems, there is an agent which interacts with an environment via performing certain actions. To this, the environment responds by providing the agent with rewards. Each action takes the environment from the present to a new state, and the agent's task is to optimize its actions so as to maximize the net reward it receives over its lifetime.

The function computing these decisions is known as policy. One of the standard ways to solve the RL problem is to first learn an optimum state–action value function $Q^*(s,a) := \mathbb{E}_{\pi^*}\left[\sum_{t=1}^{\infty} \vartheta^{t-1} r_t \mid s_0 = s\right]$, which provides the maximum expected net reward that can be achieved from the starting state s_0 by following the optimal policy π^*. Policy function can then be found out directly from this value function (Section 3.3). When the state space of the environment is large (and continuous), as is often the case with practical problems, the state value function is modeled using a function approximator, such as a linear model or a neural network: $Q_\theta(s,a)$, where θ represents the learnable parameters of the value function.

Appearing in Proceedings of the 3rd Indian Workshop on Machine Learning, IIT (BHU) Varanasi, India, 2018.

In this setting, a loss function is formulated which quantifies the goodness of the state value function in its predictions. This effectively reduces the RL problem to training this value function on the data sampled from the environment to minimize the formulated loss function. A popular form of optimization is gradient based methods.

In this work, we propose new RL algorithms which use mirror–descent (Beck & Teboulle, 2003) to learn action–value function in the robust–RL setting (Roy et al., 2017).

2. Background

2.1. Mirror Descent

The mirror descent algorithm (MDA) (Beck & Teboulle, 2003) is a gradient based optimization technique which is highly suitable for large scale settings, because its rate of convergence is weakly dependent on the dimensionality of the problem being solved.

Mirror descent can be viewed as a generalization of the simple gradient descent algorithm. Simple gradient descent works by minimizing the quadratic approximation of the function (being optimized) at the current point: $x_{t+1} = \arg\min_x \left[f(x_t) + \partial f(x_t)^\top (x - x_t) + \frac{1}{2}\|x - x_t\|_2^2\right]$. Mirror descent replaces the squared norm distance function, in the above iteration, with Bregman Divergence $B_\psi : X \times X \to \mathbb{R}^+$, defined by $B_\psi(x,y) := \psi(x) - \left[\psi(y) + \nabla\psi(y)^\top (x-y)\right]$, where ψ is a strongly convex function. Bregman divergence is a general class of distance functions and contains multiple distance functions as special cases (e.g. Squared Loss: $B_\psi(x) = (x-y)^2$ for $\psi(x) = x^2$).

Thus, the mirror descent equation is: $x_{t+1} = \arg\min_x \left[f(x_t) + \partial f(x_t)^\top (x - x_t) + B_\psi(x, x_t)\right]$, which reduces to the following equivalent Mirror Descent update:

$$x_{t+1} = \nabla\psi^{-1}(\nabla\psi(x_t) - \partial f(x_t)), \tag{1}$$

where $\nabla\psi^{-1}$ is equal to $\nabla\psi^*$, and ψ^* is the Legendre

transform of ψ defined by $\psi^*(y) := \max_{z \in C}[z^\top y - \psi(z)]$.

MDA has been previously applied to Q–learning (Mahadevan & Liu, 2012), where different p–norm functions were used as Bregman divergences. This technique lead to faster convergence rate and reduced variance for Q–learning. We aim to achieve similar results with the Robust RL framework.

2.2. Robust RL

In RL, the environment is modeled as a Markov Decision Process (MDP), with some fixed underlying transition probabilities. In Robust RL framework (Nilim & Ghaoui, 2003), these transition probabilities themselves are uncertain. They may be chosen by the environment, so as to worsen the performance of the RL agent. The goal of the agent then becomes to obtain the maximal reward from these worst case conditions, by choosing optimal actions.

Let $\mathcal{T}$ denote the space of the transition probabilities, $\mathcal{P}_\pi$ the policy space, and $c_t(i_t, a_t(i))$ denote the cost (needs to be minimized, in contrast to reward) incurred by taking action $a_t(i)$ in state i_t. Then the total expected cost (under policy π and a sequence of transition probabilities τ) is given by $C_N(\pi, \tau) := \mathbb{E}\left[\sum_{t=0}^{N-1} c_t(i_t, a_t(i)) + c_N(i_N)\right]$, where N is the episode duration and $c_N(i_N)$ is the cost of attaining the terminal state. The goal of the agent, is then to find a policy π which minimizes the cost:

$$\min_{\pi \in \mathcal{P}_\pi} \max_{\tau \in \mathcal{T}} C_N(\pi, \tau).$$

In this framework the state value function and the policy $\pi^* = (a_0^*, \ldots, a_{N-1}^*)$ can be found out as

$$v_t(i) = \min_{a \in A} \left(c_t(i, a) + \sigma_{P_i^a}(v_{t+1})\right) \tag{2}$$

$$a_t^*(i) = \arg\min_{a \in A} \left(c_t(i, a) + \sigma_{P_i^a}(v_{t+1})\right), \tag{3}$$

where $i \in X$ (set of all states), A is the set of all possible actions, $\sigma_P(v) := \sup\{p^\top v \mid p \in P\}$ is the support function of set P and P_i^a is the transition probability matrix corresponding to when the agent is in state i and has taken action a.

The above framework is extended to model–free RL case in (Roy et al., 2017). This work poses the uncertainty in transition probabilities as a model mismatch scenario, where the agent has access to only an approximate simulation of the actual environment. The perceived transition probabilities P_i^a are then modeled as

$$P_i^a := \{x + p_i^a \mid x \in U_i^a\}, \tag{4}$$

where p_i^a is the unknown true state transition probability matrix from state $i \in X$ to other states of X, given the current action a, and U_i^a is the confidence region. The confidence region itself can be chosen to be of different types such as an ellipsoid, parallelopiped, etc.

The robust setting modifies the standard TD–error. The robust TD–error is defined as :

$$\tilde{d} := c(i, \pi(i)) + \vartheta v_\theta(i') + \vartheta \sigma_{\hat{U}}(v_\theta) - v_\theta(i), \tag{5}$$

where $\hat{U}$ is the confidence interval, v_θ is the parameterized state value function, ϑ is the discount factor, i is the present state, i' is the next state and the current action is taken according to the policy π.

The Bellman operator is accordingly modified to the Robust Bellman Operator, defined by:

$$(\hat{T}_\pi v_\theta)(i) := c(i, \pi(i)) + \vartheta \sigma_{P_i^{\pi(i)}}(v_\theta). \tag{6}$$

We initially assume that v_θ is linearly parameterized and later extend to non–linear setting in Section 3.2. So, $v_\theta := \Phi\theta$, where $\Phi := [\phi_1, \phi_2, \ldots, \phi_{|X|}]$ is the set of all states (represented as features ϕ). Let Π be a transformation which projects its input function to the nearest possible linear function approximator. It is defined as

$$\Pi v := v_\theta = \Phi\theta, \text{where } \theta = \arg\min_\theta \|v_\theta - v\|_\xi^2, \tag{7}$$

where v is any general function and ξ is the stationary state distribution. Then the robust analogue of the MSPBE (Sutton et al., 2009), Mean Squared Robust Projected Bellman Error is defined as:

$$\text{MSRPBE}(\theta) = \left\|v_\theta - \Pi\hat{T}_\pi v_\theta\right\|_\xi^2 \tag{8}$$

$$= \mathbb{E}[\tilde{d}\phi]^\top \mathbb{E}[\phi\phi^\top]^{-1} \mathbb{E}[\tilde{d}\phi], \tag{9}$$

where $\hat{T}_\pi$ is the robust Bellman Operator (Eq. 6). Using these functions, we derive the MDA based robust RL algorithms for state value function in the next section.

3. Mirror Descent for Robust RL

3.1. State Value Function with Linear Function Approximator

In this section, we derive our proposed MDA based Robust RL algorithms for linearly parameterized state–value functions. We need to find θ which minimizes the MSRPB error (Eq. 8). To do this, we first find the gradient of this loss function, and then apply MDA (Eq. 1) to it.

Let $\mu_P(\theta)$ denote the gradient of $\sigma_P(v)$. Then

$$\mu_P(\theta) := \nabla_\theta \sigma_P(v_\theta) = \arg\max_{y \in \Phi^\top(P)} y^\top \theta \tag{10}$$

The gradient of the loss $-\frac{1}{2}\nabla_\theta \text{MSRPBE}(\theta)$ is

$$\begin{aligned}
&= -\mathbb{E}[\nabla_\theta(\tilde{d}\phi)]\,\mathbb{E}[\phi\phi^\top]^{-1}\,\mathbb{E}[\tilde{d}\phi] \\
&= \mathbb{E}[(\phi - \vartheta\mu_{\hat{U}}(\theta) - \vartheta\phi')\phi^\top]\,\mathbb{E}[\phi\phi^\top]^{-1}\,\mathbb{E}[\tilde{d}\phi] \\
&= \mathbb{E}[(\phi - \vartheta\mu_{\hat{U}}(\theta) - \vartheta\phi')\phi^\top]w \qquad (11) \\
&= \left(\mathbb{E}[\phi\phi^\top] - \mathbb{E}[\vartheta\mu_{\hat{U}}(\theta)\phi^\top] - \mathbb{E}[\vartheta\phi'\phi^\top]\right)\mathbb{E}[\phi\phi^\top]^{-1}\,\mathbb{E}[\tilde{d}\phi] \\
&= \mathbb{E}[\tilde{d}\phi] - \vartheta\,\mathbb{E}[\phi'\phi^\top]w - \vartheta\,\mathbb{E}[\mu_{\hat{U}}(\theta)\phi^\top]w, \qquad (12)
\end{aligned}$$

where $w = \mathbb{E}[\phi\phi^\top]^{-1}\,\mathbb{E}[\tilde{d}\phi]$, and for brevity we write the feature vector for k^{th} state: $\phi(i_k)$ as ϕ and the feature vector corresponding to the $(k+1)^{th}$ state: $\phi(i_{k+1})$ as ϕ'.

We can now give the final form of the Robust MDA update equations. We use two timescales (Sutton et al., 2009):

$$w_{k+1} = w_k + \beta_k(\tilde{d}_k - \phi_k^\top w_k)\phi_k, \tag{13}$$

with robust–MDTD updates defined as (analogous to GTD2, gradient sampled using Eq. 11)

$$\begin{aligned}
\zeta_k &= \nabla\psi(\theta_k) && (14)\\
\zeta_{k+1} &= \zeta_k + \alpha_k(\phi_k - \vartheta\phi'_k - \vartheta\mu_{\hat{U}}(\theta_k))(\phi_k^\top w_k) && (15)\\
\theta_{k+1} &= \nabla\psi^*(\zeta_{k+1}), && (16)
\end{aligned}$$

and robust–MDTDC updates defined as (analogous to TDC, gradient sampled using Eq. 12)

$$\begin{aligned}
\zeta_k &= \nabla\psi(\theta_k) && (17)\\
\zeta_{k+1} &= \zeta_k + \alpha_k(\tilde{d}_k\phi_k - (\vartheta\phi'_k + \vartheta\mu_{\hat{U}}(\theta_k))(\phi_k^\top w_k)) && (18)\\
\theta_{k+1} &= \nabla\psi^*(\zeta_{k+1}), && (19)
\end{aligned}$$

where ψ is the distance generating function, ψ^* is its Legendre transform.

3.2. State Value Function with Non–Linear Function Approximator

We will now generalize the above results to the case of non–linear value function approximator. This section closely follows the derivations given in (Maei et al., 2009).

As before, let v_θ be the value function approximator, only this time non–linear in θ. Let $\mathcal{M} := \{v_\theta \mid \theta \in \mathbb{R}^d\}$ be the manifold spanned by v_θ, then define the tangent plane of $\mathcal{M}$ at θ by $T\mathcal{M}_\theta := \{\Phi_\theta u \mid u \in \mathbb{R}^d\}$, where $\Phi_\theta(i,j) := \frac{\partial v_\theta(i)}{\partial \theta_j}$ is the Jacobian of v_θ with respect to θ. We use Π_θ to denote the projection onto the space $T\mathcal{M}_\theta$ (similar to Eq. 7). This modifies the MSRPBE(θ)

$$\begin{aligned}
&:= \left\| v_\theta - \Pi_\theta \hat{T} v_\theta \right\|_\xi^2 && (20)\\
&= \mathbb{E}[\tilde{d}\nabla v_\theta(i)]^\top\,\mathbb{E}[\nabla v_\theta(i)\nabla v_\theta(i)^\top]^{-1}\,\mathbb{E}[\tilde{d}\nabla v_\theta(i)].
\end{aligned}$$

Let $\mu_P(\theta) := \nabla_\theta\sigma_P(v_\theta) = \arg\max_{y\in\Phi^\top(P)} y^\top\theta$ and define

$$h(\theta, u) := -\mathbb{E}\left[(\tilde{d} - \phi^\top u)\nabla^2 v_\theta(i) u\right]. \tag{21}$$

Then the gradient of the loss $-\frac{1}{2}\nabla_\theta\text{MSRPBE}(\theta) = \mathbb{E}[(\phi - \vartheta\mu_{\hat{U}}(\theta) - \vartheta\phi')\phi^\top]w + h(\theta, w)$. Note that, here $\phi \equiv \nabla v_\theta(i)$ and $\phi' \equiv \nabla v_\theta(i')$.

We can now give the algorithms for the non–linear setting:

$$\begin{aligned}
w_{k+1} &= w_k + \beta_k(\tilde{d}_k - \phi_k^\top w_k)\phi_k && (22)\\
h_k &= (\tilde{d}_k - \phi_k^\top w_k)\nabla^2 v_{\theta_k}(i_k)\, w_k, && (23)
\end{aligned}$$

with robust–MDTD update given by

$$\begin{aligned}
\zeta_k &= \nabla\psi(\theta_k)\\
\zeta_{k+1} &= \zeta_k + \alpha_k\{(\phi_k - \vartheta\phi'_k - \vartheta\mu_{\hat{U}}(\theta_k))(\phi_k^\top w_k) - h_k\}\\
\theta_{k+1} &= \nabla\psi^*(\zeta_{k+1}),
\end{aligned}$$

and robust–MDTDC update given by

$$\begin{aligned}
\zeta_k &= \nabla\psi(\theta_k)\\
\zeta_{k+1} &= \zeta_k + \alpha_k\{\tilde{d}_k\phi_k - (\vartheta\phi'_k + \vartheta\mu_{\hat{U}}(\theta_k))(\phi_k^\top w_k) - h_k\}\\
\theta_{k+1} &= \nabla\psi^*(\zeta_{k+1}).
\end{aligned}$$

3.3. Control and Learning of State–Action Value Function

The algorithms given above, estimate the state value function $V(s)$. However, obtaining a policy from $V(s)$ is not straightforward. Therefore, the algorithms need to be modified in a minor way (Maei et al., 2010) by learning the state–action value function $Q(s,a)$ instead of $V(s)$.

We model the Q function (in case of a linear function approximator) as $Q_\theta(s,a) = \phi_{s,a}\theta$, where $\phi_{s,a}$ represents the features composed of both the actions taken in a state and that state itself. Then, Eq. 15 and Eq. 18 would use $\phi_k = \phi(s_k, a_k)$, and $\phi'_k = \phi(s_{k+1}, \pi(s_{k+1}))$ (in case of Greedy policy such as $\pi(s) = \arg\max_a Q_\theta(s,a)$) or $\phi'_k = \phi(s_{k+1}, a'_{k+1}) + Q_\theta(s_{k+1}, a'_{k+1})\nabla\ln\pi(a'_{k+1}|s_{k+1})$, with a'_{k+1} sampled from $\pi(\cdot|s_{k+1})$ (in case of a stochastic policy such as Gibbs policy: $\pi(a|s) = \frac{e^{Q_\theta(s,a)}}{\sum_b e^{Q_\theta(s,b)}}$). A similar modification should work in the non–linear setting as well.

4. Conclusions and Future Work

In this work, we proposed novel TD–learning algorithms for Robust–RL using function approximators, which use mirror descent as optimizer. We also showed ways to extend the algorithms to control tasks. Mirror Descent has properties which make it ideal for application to RL, like its rate of convergence which weakly depends on the dimensionality of the problem. Whereas, Robust RL framework makes relaxing assumptions on the knowledge of the environment transition probabilities.

Through our work, we aim to bring the properties of mirror descent to robust RL framework. We will implement these algorithms on various control tasks and compare them with their gradient descent counterparts. We'll also investigate their theoretical properties.

Acknowledgment

The author would like to thank Dr. K. Lakshmanan for his constant support and technical assistance throughout this work, and the anonymous reviewer and the Editor for their comments.

References

Beck, Amir and Teboulle, Marc. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Oper. Res. Lett.*, 31(3):167–175, 2003. doi: 10.1016/S0167-6377(02)00231-6. URL https://doi.org/10.1016/S0167-6377(02)00231-6.

Maei, Hamid Reza, Szepesvári, Csaba, Bhatnagar, Shalabh, Precup, Doina, Silver, David, and Sutton, Richard S. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pp. 1204–1212, 2009. URL http://papers.nips.cc/paper/3809-convergent-temporal-difference-learn/ing-with-arbitrary-smooth-function-appro/ximation.

Maei, Hamid Reza, Szepesvári, Csaba, Bhatnagar, Shalabh, and Sutton, Richard S. Toward off-policy learning control with function approximation. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pp. 719–726, 2010. URL http://www.icml2010.org/papers/627.pdf.

Mahadevan, Sridhar and Liu, Bo. Sparse q-learning with mirror descent. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012*, pp. 564–573, 2012. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2317&proceeding_id=28.

Nilim, Arnab and Ghaoui, Laurent El. Robustness in markov decision problems with uncertain transition matrices. In *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pp. 839–846, 2003. URL http://papers.nips.cc/paper/2367-robustness-in-markov-decision-proble/ms-with-uncertain-transition-matrices.

Roy, Aurko, Xu, Huan, and Pokutta, Sebastian. Reinforcement learning under model mismatch. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 3046–3055, 2017. URL http://papers.nips.cc/paper/6897-reinforcement-learning-under-model-m/ismatch.

Sutton, Richard S., Maei, Hamid Reza, Precup, Doina, Bhatnagar, Shalabh, Silver, David, Szepesvári, Csaba, and Wiewiora, Eric. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, pp. 993–1000, 2009. doi: 10.1145/1553374.1553501. URL http://doi.acm.org/10.1145/1553374.1553501.