Bootstrapped Gradient Temporal-Difference Learning

Dominik Meyer[1] Martin Knopp[1] Hao Shen[1]

***Abstract*— In this work we aim at providing a overview on gradient based temporal difference learning methods in reinforcement learning. We will look at three different cost functions, the mean squared Bellman error, the mean squared projected Bellman error and the norm of the expected update. Finally we will derive two new on-line gradient algorithms for TD learning, that base on the idea of bootstrapping.**

***Index Terms*— Reinforcement Learning (RL), Stochastic Gradient Descent, Bootstrapping, Gradient Temporal-Difference (GTD)**

I. Introduction

Reinforcement Learning (RL) is a machine learning technique, which is especially well suited to solve control problems in systems, where sequential actions have to be taken. The basis for RL, Markov Decision Processes and Dynamic Programming have been well studied in the fields of engineering and operations research. Since the field of Artificial Intelligence (AI) picked up these topics, alternative and approximative solution methods have been contributed to this field.

Here we consider learning to take place in the framework of a Markov Reward Process (MRP). It consists of a tuple $(\mathscr{S}, P, R, \gamma)$, where $\mathscr{S}$ is the set of possible state of the environment, $P : \mathscr{S} \times \mathscr{S} \to [0,1]$ the conditional transition probabilities $P(s,s')$ to go from state s to s', $R : \mathscr{S} \to \mathbb{R}$ a reward function, assigning immediate reward r to a state s and $\gamma \in [0,1]$ a discount factor.

One central goal in RL is the estimation of the so called value function, which is the expectation of future reweards

$$V : \mathscr{S} \to \mathbb{R} \qquad V(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s\right] \tag{1}$$

for all states of the environment.

It is known, that the Bellman equation holds for the value function

$$V(s) = R(s) + \gamma \sum_{s'} P(s,s') V(s'). \tag{2}$$

The right hand side of equation (2) is often referred to as the Bellman operator, denoted as $\mathscr{T}V(s)$. The value function $V(s)$ is a fixed point of the Bellman operator $\mathscr{T}V(s)$, i.e. $V(s) = \mathscr{T}V(s)$.

In may application scenarios, the state space is to large to be represented by a simple table. Function approximation is therefore a valuable tool to enable learning in such scenarios. A very common approach is to construct a set of features $\phi : \mathscr{S} \to \mathbb{R}^k$ on the perceived states and then approximate the value function with a linear model. For a given state the value function will be approximated by

$$V(s) \approx (\phi(s))^\top \theta =: V_\theta \in \mathscr{F}, \tag{3}$$

where $\theta \in \mathbb{R}^k$ is a parameter vector and $\mathscr{F} := \{\Phi\theta | \theta \in \mathbb{R}^k\}$ is the hypothesis space of possible value function approximations, i.e. the span of the features $\Phi := \phi(\mathscr{S})$. As a shorthand we will write ϕ and ϕ' instead of $\phi(s)$ and $\phi(s')$, respectively.

II. Derivation of Gradient Temporal Differences

In the setting of on-line Temporal Difference (TD) learning, tuples (s_t, r_t, s'_t) of state transitions are sampled and the parameter vector θ is updated at each time step t.

This formulation allows us to formulate the problem as a supervised learning problem, where we aim at finding the parameter vector θ. By using the fixed point property $V = R + \gamma PV = \mathscr{T}V$ of the Bellman operator the Mean Square Bellman Error is defined as

$$\begin{aligned} J_1 : \mathbb{R}^k \to \mathbb{R} \qquad J_1(\theta) &:= \tfrac{1}{2}\|V_\theta - \mathscr{T}V_\theta\|^2 \\ &= \tfrac{1}{2}(\mathbb{E}[\delta_\theta])^2 \qquad \text{(MSBE)}, \end{aligned} \tag{4}$$

with $\delta_\theta = \delta(\theta) := r + \theta^\top(\gamma\phi' - \phi)$ being the TD error. Unfortunately, $\mathscr{T}V_\theta$ might not lie in $\mathscr{F}$, but can be projected in to the hypothesis space by the projector $\Pi = \Phi^\top \left(\Phi\Phi^\top\right)^{-1}\Phi$.

This leads to our second objective function

$$\begin{aligned} J_2 : \mathbb{R}^k \to \mathbb{R} \qquad J_2(\theta) &:= \tfrac{1}{2}\|V_\theta - \Pi\mathscr{T}V_\theta\|^2 \\ &= \tfrac{1}{2}\mathbb{E}[\delta_\theta\phi]^\top \mathbb{E}[\phi\phi^\top]^{-1}\mathbb{E}[\delta_\theta\phi] \qquad \text{(MSPBE)}, \end{aligned} \tag{5}$$

which is called the Mean Squared Projected Bellman Error.

Recall, that in TD(0) learning with linear function approximation the parameter vector is updated as $\theta_{t+1} = \theta_t + \alpha_t \delta_{\theta_t}\phi$, where $\alpha_t > 0$ is a sequence of step size parameters. The last part $\mathbb{E}[\delta_\theta\phi] \in \mathbb{R}^k$ can be regarded as the remaining error for a given θ. This should be zero and we can therefore derive a third cost function

$$\begin{aligned} J_3 : \mathbb{R}^k \to \mathbb{R} \qquad J_3(\theta) &:= \tfrac{1}{2}\|\mathbb{E}[\delta_\theta\phi]\|^2 \\ &= \tfrac{1}{2}\mathbb{E}[\delta_\theta\phi]^\top \mathbb{E}[\delta_\theta\phi] \qquad \text{(NEU)}, \end{aligned} \tag{6}$$

which is called the Norm of the Expected Update.

To minimize the above three cost functions, we employ a stochastic gradient descent algorithm. Generally this can be denoted by

$$\theta_{t+1} = \theta_t + \alpha_t \nabla_\theta J_i(\theta_t). \tag{7}$$

Therefore we need to derive the gradients to our three objective functions.

*This work was partly supported by the International Graduate School of Science and Engineering, TUM

[1]The authors are with the Institute for Data processing, Technische Universität München, Germany. <firstname.lastname>@tum.de

The TD error δ_θ contains two occurences of the parametrized value function, one for the state s and one for the state s' to which we transited to. If we now introduce two independent parametrizations θ_1 and θ_2, we arrive at a TD error like

$$\delta_{\theta_1,\theta_2} = r + \gamma\theta_1^\top\phi' - \theta_2^\top\phi. \tag{8}$$

In each case we can choose to set $\theta_1 = \theta_2 = \theta$ and derive with respect to θ, or consider $\theta_1 = \theta_{t-1}$ being fixed and derive with respect to $\theta_2 = \theta$. The latter is called bootstrapping, where we use estimates of the value function with informations from the previous iteration.

Let us illustrate the difference at the example of two very well known algorithms, which can be derived from the first cost function J_1. If we now choose to set $\theta_1 = \theta_2 = \theta$ and derive the gradient, we arrive at

$$\nabla J_1(\theta) = \mathbb{E}[\delta_\theta(\gamma\phi' - \phi)], \qquad \text{(BRM)} \tag{9}$$

which is the Bellman Residual Minimization [4]. On the other hand, with setting $\theta_1 = \theta_{t-1}$ and $\theta_2 = \theta$, we can derive

$$\nabla J_1(\theta) = -\mathbb{E}[\delta_\theta\phi], \qquad \text{(TD)} \tag{10}$$

which is the original Temporal Difference algorithm for linear value function approximation [3].

The same can be done with the third cost function, and by deriving the gradient without using bootstrapping, we get

$$\nabla J_3(\theta) = (\mathbb{E}[(\gamma\phi' - \phi)\phi^\top])^\top\mathbb{E}[\delta_\theta\phi], \qquad \text{(GTD)} \tag{11}$$

which is the recently introduced Gradient Temporal Difference learning algorithm [1], which is known to be convergent in off-policy settings with function approximation, which unfortunately does not hold for BRM.

If we now introduce bootstrapping to the third cost function and derive the gradient, we get an update in the form of

$$\nabla J_3(\theta) = -\mathbb{E}[\phi\phi^\top]\mathbb{E}[\delta_\theta\phi], \qquad \text{(BS-GTD)} \tag{12}$$

which we will call Bootstrapped Gradient Temporal Difference learning. These two gradient updates unfortunately have the property to be biased, if calculated with only one sample at a time. In BRM, therefore, double sampling was introduced, where for each update, two independent samples have to be collected. Of course, this is only possible, if we have a model or simulator of the learning process at hand. In [1] the authors therefore introduced a trick, to avoid this double sampling. They introduced a set of secondary weights, which will be used to estimate one of the expectation terms in the gradient update

$$w :\approx \mathbb{E}[\delta_\theta\phi]. \tag{13}$$

The gradient updates for the third cost function, therefore consist of

$$\begin{aligned} \nabla J_3(\theta) &= (\mathbb{E}[(\gamma\phi' - \phi)\phi^\top])^\top w, && \text{(GTD)} \\ \nabla J_3(\theta) &= -\mathbb{E}[\phi\phi^\top]w. && \text{(BS-GTD)} \end{aligned} \tag{14}$$

For the second cost function, we arrive at analogous results and can obtain the Gradient Temporal Difference 2 (GTD2) algorithm [2] and a bootstrapped version of it

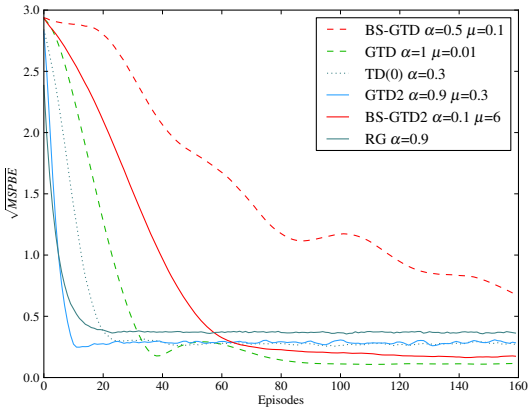
$$\begin{aligned} \nabla J_2(\theta) &= (\mathbb{E}[(\gamma\phi' - \phi)\phi^\top])^\top(\mathbb{E}[\phi\phi^\top])^{-1}\mathbb{E}[\delta_\theta\phi] \\ &\approx \mathbb{E}[(\gamma\phi' - \phi)\phi^\top]u, && \text{(GTD2)} \\ \nabla J_2(\theta) &= -\mathbb{E}[\phi\phi^\top](\mathbb{E}[\phi\phi^\top])^{-1}\mathbb{E}[\delta_\theta\phi] \\ &= -\mathbb{E}[\delta_\theta\phi] \\ &\approx -\mathbb{E}[\phi\phi^\top]u. && \text{(BS-GTD2),} \end{aligned} \tag{15}$$

with $u \approx (\mathbb{E}[\phi\phi^\top])^{-1}\mathbb{E}[\delta_\theta\phi]$.

Two things are here to notice: First, we also need a set of secondary weights, to not have to double sample. Second, it is to notice, that for the bootstrapped version of GTD2, we either arrive at the same update, as classical TD learning, or by approximating again two of the expectations we arrive at the same update, as we already had for BS-GTD. Therefore, we have only gained new insight into the derivation of those two algorithms, and their relation to the MSPBE.

III. Empirical Results

We compared the new bootstrapped versions to their non-bootstrapped counterparts with respect to their empirical convergence. All experiments were run on a 14 state "Boyan chain" with 4 features. Further details about this MRP environment can be found in [5]. The learning rates were fixed through the whole learning process and if there were two learning rate parameters, their relation was $\alpha = \mu \cdot \beta$.



References

[1] R. S. Sutton, Csaba Szepesvári, and H. R. Maei. A convergent $O(n)$ algorithm for off-policy temporal-difference learning with linear function approximations. In *NIPS 21*, pages 1609–1616. The MIT Press, 2008.

[2] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of ICML 2009*, pages 993–1000, 2009.

[3] R. S. Sutton, and A. G. Barto. Reinforcement Learning: An Introduction, Cambridge, MA, USA: MIT Press, March, 1998.

[4] L. C.Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of ICML 1995* pages 30-37, 1995.

[5] J. Boyan. Technical update: Least-squares temporal difference learning. In *Machine Learning*, 49:233–246, 2002.