Instructions for paper submissions to AISTATS 2017

Anonymous Author 1
Unknown Institution 1

Anonymous Author 2
Unknown Institution 2

Anonymous Author 3
Unknown Institution 3

Abstract

1 Introduction

Reinforcement Learning is a paradigm where the learning algorithm/agent needs to learn via direct interaction with the environment. In most cases, the underlying environment can be modelled as a Markov Decision Process (MDP). At time any given time instant, the agent knows the state (described via the state variables) of the environment (i.e., the MDP) and has to choose an action from a given action set. The action selection mechanism of the agent is formally known as the policy, and each policy is associated with a value function, which is a mapping from the set of states to reals that specifies the value obtained by following a given policy starting from a given state. Typically, in order to improve a given policy, its value function needs to be computed first. Thus learning the value function of a given policy assumes centre stage.

Algorithms that learn the value function need to address two important issues namely lack of explicit model information (of the underlying MDP) and dimensionality of the state space. It is known that the value function of a policy can be computed by solving a $|S| \times |S|$ (where $|S|$ is the cardinality of the state space) linear system known as the Bellman equation (BE). However, the coefficients of the BE are described in terms of the model parameters, which are not known explicitly in an RL setting. Due to the *curse-of-dimensionality* $|S|$ grows exponentially in the number of state variables. As a result, when $|S|$ is very large, it is infeasible to store, retrieve and compute the value for each and every state.

Linear Function Approximation (LFA) is a common dimensionality reduction approach employed to reduce the computational overhead when $|S|$ is large. In this approach, the value function is approximated by a linear combination of n ($<< |S|$) basis functions. Once the basis is chosen, an approximate value function can computed by solving a reduced $(n \times n)$ linear system known as the projected Bellman equation (PBE). It is important to note that the coefficients of the PBE are dependent not only on the basis functions but also on the model parameters. Thus, RL algorithms need to compute the PBE solution using only the samples obtained via direct interaction.

The temporal difference (TD) family of learning algorithms are a very important sub-class of RL algorithms and compute the solution to the PBE. The first TD algorithm (i.e., TD(0)) was introduced in [] and TD with LFA was analyzed in []. The TD family includes several variants such as TD such as TD(λ), SARSA, Q-learning, LSTD, iLSTD and Gradient-TD (GTD) []. Aspects such as sample efficiency, computational complexity and stability have driven the course of development of these various TD algorithms. The issue with TD(0) is that it makes use of only the current sample and is not sample efficient. This issue was alleviated by the LTSD by making use of all the data to estimate and solve an empirical PBE. Since LSTD involves matrix updates and inversions at each step its computational complexity is minimum $O(n^2)$. In contrast, since simple TD(0) involves only $O(n)$ computations per step, it continues to be of interest in applications in which cannot afford $O(n^2)$ required by LSTD.

The TD, Q-learning and SARSA algorithms are not stable in the most general setting, especially the *off-policy* case. Here, the policy (*target*) whose value function needs to be computed which is different from the policy (*behaviour*) that was used to collect the samples. The instability arises due to the fact that TD, SARSA and Q-learning are not true gradient algorithms. The Gradient TD (GTD) family of algorithms perform updates in the gradient direction and are stable. Even though GTD is stable in the off-policy case, it was pointed out in [], that even the GTD is not a true stochastic gradient descent algorithm with respect to

Preliminary work. Under review by AISTATS 2017. Do not distribute.

its original objective []. Authors in [] propose newer variants of the GTD and show that they are true gradient algorithms, however, with respect to a different objective function obtained via a primal-dual saddle point formulation.

Motivation The focus of this paper is the TD family of algorithms and we now describe our motivation. Most TD algorithms (except the LSTD) perform incremental updates, and an important aspect that affects the speed of convergence is the choice of step size. While, there are adaptive step-size that helps in increasing rates of convergence, a clear cut step size rule is missing in literature. Even though all the TD variants deal with the PBE, the derivation of the various algorithms are rather ad-hoc []. As a result, there is no holistic framework that enables us to design stable and computationally efficient RL algorithms.

Methodology Most TD algorithms perform noisy updates to the weights using at time step, and hence under the category of stochastic approximation algorithms. The term stochastic approximation signifies the fact that the noisy quantities used in the algorithms are stochastically approximate, and are equal to the quantities only in the expectation. The SA oft theory considers the algorithm to exhibit an 'average behaviour which is corrupted by noise. Under certain reasonable conditions, the noise can be discarded in the asymptotic limit and the average behaviour can then be shown to be well approximated by an ordinary differential equation corresponding to a given SA algorithm. The asymptotic behaviour of the algorithm can then studied by analyzing the ODE. In this paper, we make use of the theory of stochastic approximation algorithms to understand the TD algorithms in a holistic manner.

The specific contributions are listed as under.

- We then argue on the lines of [] and show that the finite time performance of any TD algorithm has two terms namely the bias term (due to the initial conditions) and the variance terms (due to the noisy samples).
- An interesting feature to note is that the bias term is affected by the condition number of the matrix involved in the PBE. This is in line with the asymptotic results in SA theory, i.e., the driving matrix of the ODE corresponding to the algorithm affects its asymptotic rate of convergence.
- We argue that the ill-effects of conditioning can be handled successfully by using constant step sizes and iterate averaging (state the order etc). This eliminates the fuzziness around the choice of step sizes.
- The SA approach enable us to view TD algorithms as discretization of ODEs that converge to the PBE solution. Using this view, we also derive a new and stable TD algorithm.

We also present numerical examples along side the analysis.

2 Background

We now present a brief overview of Markov Decision Processes (MDPs) which is a framework to describe the RL setting, i.e., the agent's interaction with the environment. An MDP is a five tuple and is denoted by $M = \{S, A, P, R, \gamma\}$, where S is the state space, A is the action space, $P = (p_a(s, s'), a \in A, s, s \in S)$ is the probability transition kernel that specifies the probability of transitioning from state s to s' when the agent taking an action a, and $R(s, a) \colon S \times A \to \mathbb{R}$ is the reward for taking an action a in state s and $0 < \gamma < 1$ is a given discount factor. A stationary deterministic policy (or simply a policy) is a denoted by $\pi = (\pi(s, \cdot), s \in S)$, where $\pi(s, \cdot)$ is a probability distribution over the set of actions. The infinite horizon discounted reward of a policy π is given by

$$J_\pi = \mathbf{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, \pi], \tag{1}$$

and is the discounted sum of rewards starting from state s and taking actions according to policy π.

It is of interest to compute the optimal policy π^* which yields the maximum reward starting from any given state. A key step in computing the optimal policy is *policy evaluation*, i.e., computing J_π of a given policy π.

2.1 Bellman Equation

The value function J_π can be computed by solving the Bellman equation (BE) given below:

$$J_\pi = T_\pi J_\pi, \tag{2}$$

where in (4) $T_\pi \colon \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ is the Bellman operator. The Bellman operators is defined in terms of the model parameters of the MDP as follows

$$(T_\pi J)(s) = \sum_{a \in A} \pi(s, a)(R(s, a) + \gamma \sum_{s \in S} p_a(s, s) J(s))), \tag{3}$$

where $J_\pi = (J_\pi(s), \forall s \in S)$. The BE can thus be re-written as

$$J_\pi = R_\pi + \gamma P_\pi J_\pi, \tag{4}$$

where $R_\pi(s) = \sum_{a\in A}\pi(s,a)R(s,a)$ and $p_\pi(s,s) = \sum_{a\in A}\pi(s,a)p_a(s,s)$. Note that the BE is a linear system of equations in $|S|$ variables.

2.2 Function Approximation

The term *curse-of-dimensionality* denotes the fact that $|S|$ scales exponentially in the number of state variables. Due to the curse, MDPs in practice tend to have large number of states. Solving the linear system in (4) involves matrix inversion and requires a minimum of $O(|S|^2)$ computations, and hence it is impractical to solve the BE when $|S|$ is large. A practical way to address this issue is to approximate the value function. A widely adopted approach is function approximation, wherein the value function is approximated by a parameterized family of functions $J_\pi \approx J_\theta = \Phi(\theta^*)$, where $\{\Phi(\theta), \theta \in \mathbb{R}^n\}$ is the parameterized family and $\theta^* \in \mathbb{R}^n$ is the optimal parameter. Note that the parameter θ belongs to $\mathbb{R}^n$ and dimensionality reduction can be achieved by choosing $n << |S|$. Linear parameterization is the most common, wherein, $J_\pi \approx J_\theta\Phi\theta^*$, with Φ as a $|S| \times n$ feature matrix.

2.3 Projected Bellman Equation

Once Φ is fixed, we have to come up with a procedure to compute out θ^*. An immediate idea is to take a cue from linear regression and look at the following quadratic loss function that quantifies the approximation error.

$$L(\theta) = \sum_{s\in S}(J_\theta - TJ_\theta)^2. \tag{5}$$

The above penalty function is called *mean-square-Bellman-error* (MSBE) and a way to compute θ^* would be to minimize the MSBE. A caveat however is that TJ_θ does not usually belong to the subspace belonging to spanned by the columns of Φ. As a result, TJ_θ cannot be expressed as $\Phi\theta$ for any $\theta \in \mathbb{R}^n$. Due to this reason, none of the TD algorithms consider the MSBE as the error criterion.

A way to side step the issue of non-representability of TJ_θ, we can instead consider a projected Bellman equation (PBE) given by

$$\begin{aligned} J_{\theta^*} &= \Pi T_\pi J_{\theta^*}, \\ \Phi\theta^* &= \Pi T_\pi \Phi\theta^*, \end{aligned} \tag{6}$$

where $\Pi = \Phi(\Phi^\top D_\pi \Phi)^{-1}\Phi^\top$ is the least squares projection operator and D_π is a diagonal matrix with the diagonal entry $D_\pi(s,s)$ as the stationary distribution of state s under policy π.

3 Stochastic Approximation

A typical single timescale stochastic recursion is given by

$$x_{t+1} = x_t + \alpha_t(f(x_t) + M_{t+1}), \tag{7}$$

where $x_t \in \mathbb{R}^n, \forall t \geq 0$, $f\colon \mathbb{R}^n \to \mathbb{R}^n$ is a Lipschitz continuous function and $M_{t+1}, t \geq 0$ are martingale difference noise terms. Under reasonable conditions, we can analyze the iterates in (7) by studying its average behaviour. To this end, in addition to Lipschitz continuity of f, we need following assumptions.

Assumption 1.

1. $\alpha_t \to 0, t \to \infty$, $\sum_{t\geq 0}\alpha_t = \infty$ *and* $\sum_{t\geq 0}\alpha_t^2 < \infty$.
2. $\{M_t\}$ *is a martingale difference sequence with respect to an increasing family of* σ*-fields* $\mathcal{F}_t \doteq \sigma(x_0, M_1, \ldots, M_t), t \geq 0$, *such that* $\mathbf{E}[M_{t+1}|\mathbf{F}_t] = 0 a.s., t \geq 0$.. *Further,* $\{M_t\}$ *are square-integrable i.e.,* $\mathbf{E}[M_{t+1}|\mathbf{F}_t] \leq K(1+ \parallel x_t \parallel^2), a.s., \forall t \geq 0$.
3. $\sup_t ||x_t||| < \infty$
4. *The ordinary differential equation (ODE)* $\dot{x}(t) = f(x(t))$*, has a globally unique asymptotically stable equilibrium* $x^* \in \mathbb{R}^n$.

Equipped with the above assumptions we state an useful result from []

Theorem 1. *Under the above assumptions we have* $x_t \to x^*$ *as* $t \to \infty$.

4 Temporal Difference Learning

The *temporal difference* (TD) class of learning algorithms solve the projected Bellman equation. We now present the those members of the TD family of algorithms which make incremental updates to the weight vector at each iteration[1]. We also present the motivation behind the derivation of the various update rules. In what follows $\{\alpha_t \geq 0, \forall t \geq 0\}$ is a sequence of non-negative step-sizes and $\theta_t \in \mathbb{R}^n \forall t \geq 0$ denotes the weight vector at time t.

The $TD(0)$ algorithm is given by the following update rule

$$\theta_{t+1} = \theta_t + \alpha_t(\underbrace{R(s_t,a_t) + \gamma\phi(s_{t+1})\theta_t - \phi(s_t)\theta_t}_{\delta_t})\phi(s_t)^\top, \tag{8}$$

where $\phi(s_t)^\top \in \mathbb{R}^n$ is the feature of state s_t and is the s^{th} row of the feature matrix Φ. In (9), δ_t is the

[1]We do not discuss the LSTD. However, we discuss iLSTD which fits our criterion of incremental updates

temporal difference term which is an estimate of the error in the value function specified by θ_t. The $TD(0)$ update can also be re-written as

$$\theta_{t+1} = \theta_t + \alpha_t(b_\pi - A_\pi)\theta_t + \alpha_t M_{t+1} \tag{9}$$

where $b_\pi = \Phi^\top D_\pi R_\pi$ and $A_\pi = Phi^\top D_\pi(I - \gamma P_\pi)\Phi$. In order to arrive at (9) we have made use of the fact that $s_t \sim D_\pi, a_t \sim \pi(s_t, \cdot)$, $\mathbf{E}[\delta_t|s_t] = \big(R_\pi(s_t) + \gamma(P_\pi J_\theta)(s_t) - J_\theta(s_t)\big)$ and letting $M_{t+1} = \delta_t\phi(s_t)^\top - \mathbf{E}[\delta_t\phi(s_t)^\top]$.

As a consequence of Theorem 1 we have the following convergence result for TD.

Theorem 2 (Convergence of TD)**.** *The $TD(0)$ algorithm in* (9) *converges to the solution of the PBE when A_π has all its Eigen values with positive real parts.*

4.1 Off-policy $TD(0)$

The $TD(0)$ can exhibit divergent behaviour in the *off-policy* setting. In the *off-policy* setting the recursion in (9) has to be modified as follows

$$\theta_{t+1} = \theta_t + \alpha_t(\rho_t(s_t, a_t)R(s_t, a_t) + \gamma\phi(s_{t+1})\theta_t - \phi(s_t)\theta_t)\phi(s_t)^\top \tag{10}$$

where $\rho_t \doteq \frac{\pi(s_t,a_t)}{\mu(s_t,a_t)}$ is the importance sampling ratio between the behaviour policy μ and target policy π. The recursion in (10) can be re-written as

$$\theta_{t+1} = \theta_t + \alpha_t(b_\pi - A_\pi^\mu)\theta_t + \alpha_t M_{t+1}, \tag{11}$$

where $A_\pi^\mu = \Phi^\top D_\mu(I - \gamma P_\pi)\Phi$, where μ is the behaviour policy used to sample the data. The undoing of $TD(0)$ in the off-policy is due to the fact that all the eigen values of A_π^μ cannot be guaranteed to have positive real parts. This condition can be demonstrated even in simple example as discussed below

Example 1.

5 Gradient Temporal Difference Learning

The instability of $TD(0)$ is due to the fact that it is not a true gradient descent algorithm. The first gradient-TD (GTD) algorithm was proposed in [] and is based on minimizing the *norm of the expected TD update* (NEU) given by

$$NEU(\theta) = \mathbf{E}[\rho\phi^\top\delta(\theta)]\mathbf{E}[\rho\phi^\top\delta(\theta)] \tag{12}$$

The GTD scheme based is on the gradient of the above expression which is given by $-\frac{1}{2}\nabla NEU(\theta) = \mathbf{E}[\rho(\phi - \gamma\phi)^\top\phi]\mathbf{E}[\rho\phi^\top\delta(\theta)]$. Since the gradient is a product of two expectation we cannot use a sample product (due to the presence of correlations). The GTD addresses this issue by estimating $\mathbf{E}[\rho\delta\phi^\top]$ in a separate recursion. The GTD updates can be given by

$$\begin{aligned} y_{t+1} &= y_t + \alpha_t(\rho_t\phi^\top\delta_t - y_t) \\ \theta_{t+1} &= \theta_t + \alpha_t\rho_t(\phi(s_t) - \gamma\phi(s_{t+1}))^\top\phi(s_t)y_t \end{aligned} \tag{13}$$

Instead of NEU, the *mean-square projected Bellman Error* (MSPBE) can also be minimized. The MSPBE is defined as

$$MSPBE(\theta) = \| J_\theta - \Pi T_\pi J_\theta \|_D^2 \tag{14}$$

The GTD2 algorithm was proposed in [] based on minimizing (14). The GTD2 updates are given by

$$\begin{aligned} y_{t+1} &= y_t + \beta_t\phi(s_t)^\top(\rho_t\delta_t - \phi(s_t)y_t) \\ \theta_{t+1} &= \theta_t + \alpha_t\rho_t(\phi(s_t) - \gamma\phi(s_{t+1}))^\top\phi(s_t)y_t \end{aligned} \tag{15}$$

5.1 Saddle Point Formulation

6 Finite Time Analysis

We now present the finite time analysis for the family of TD algorithms. Consider the following stochastic recursion

$$x_{n+1} = x_t + \alpha_t(g - Hx_t) + \alpha_t M_{t+1} \tag{16}$$

From results of **??** we know that the above recursion converges to $x^* = H^{-1}g$. We now will study the finite time performance of the above recursion

$$\begin{aligned} x_{t+1} - x^* &= x_t + \alpha_t(g - Hx_t) + \alpha_t(M_{t+1}) - H^{-1}g \\ &= (I - \alpha_t H)(x_t - x^*) + \alpha_t M_{t+1} \\ &= \underbrace{\prod_{i=0}^{t}(I - \alpha_i H)(x_0 - x^*)}_{\text{Bias}} + \underbrace{\sum_{i=0}^{t}\prod_{i+1\le k<t}(I - \alpha_k H)\alpha_i M_{i+} \end{aligned}$$

6.1 Rupert-Polyak Averaging with constant step size

Let $\alpha_t = \alpha$ in (16) and then consider $\bar{x}_t = \frac{1}{t+1}\sum_{k=0}^{t} x_k$, we then have

$$\bar{x}_t - x^* = \frac{1}{t+1}\Big(\sum_{i=0}^{t}(I - \alpha H)^{t-i}(x_0 - x^*) + \alpha\sum_{i=0}^{t}\sum_{k=0}^{t-i}(I - \alpha H)^{t-}$$

Let $\bar{\lambda}$ be the eigen value of $(I - \alpha H)$ with maximum modulus, then we have

$$\mathbf{E}[\| \bar{x}_t - x^* \|^2] \le \frac{1}{(t+1)^2}\frac{1}{(1-\bar{\lambda})^2}\Big(\| (x_0 - x^*) \|^2 + \alpha^2\sum_{i=0}^{t}\mathbf{E} \| M \tag{17}$$

- When all eigen values of H have positive real parts, then for diminishing step sizes (16) tracks the ODE $\dot{x}(t) = (g - Hx(t))$. Suppose the eigen value of H with smallest real part, say μ had no imaginary component, then it dictates the asymptotic rate of the ODE. Similarly, the modulus of the eigen value dictates the rate at which the bias is forgotten in (17). Further, in the case when the eigen value of H with smallest real part, say μ had no imaginary component, then the rate in (17) is then dictated by $\bar{\lambda} = (I - \alpha\mu)$.

- Given that the rates with respect to time are the same for all the stochastic approximation schemes of the form (16), algorithms can be designed using the following criteria:

  1. Designing H and g so that $H^{-1}g$ is equal to the TD solution.
  2. Designing a desirable value of $\bar{\lambda}$ since it plays a role both in the bias as well as noise terms.

6.2 Optimal Step Size

7 Numerical Illustration

In this section, we compute the H matrix for the various TD algorithms discussed so far and make qualitative and quantitative comments. In order to keep the discussion simple and intuitive, we compute explicit values for the most trivial MDP namely single state MDP.

8 A new stable TD algorithm

We now modify the predictor corrector algorithm as follows

$$x_t^m = x_t + \beta(g - Hx_t) \tag{18}$$
$$x_{t+1} = x_t + \alpha_t(g - Hx_t^m) \tag{19}$$

The above recursion can then be written as

$$x_{t+1} = (I - \alpha_t(H + \beta H^2))x_t + \alpha_t(g - \beta Hg) \tag{20}$$

Notice that the matrix $H + \beta H^2$ can always be made have all real values positive.

Example 2.

9 Predictor Corrector Algorithm

Consider the ODE given by

$$\dot{x}(t) = (g - Hx(t)) \tag{21}$$

with initial condition $x(0) = x_0$. The Euler discretization of the above ODE leads to the following update rule

$$x_{t+1} = x_t + \alpha_t(g - Hx_t). \tag{22}$$

The *Predictor-Corrector* discretization of the ODE is the following update rule

$$x_t^m = x_t + \alpha_t(g - Hx_t) \tag{23}$$
$$x_{t+1} = x_t + \alpha_t(g - Hx_t^m) \tag{24}$$

The idea behind the PC method is to first take a step in the gradient direction to produce a new point x_t^m, and then obtain the estimate of the gradient at x_t^m to be used to update x_t. It is clear that the GTD-MP is using the PC type update. Notice that the PC updates can be unfurled and written as the following single recursion

$$x_{t+1} = x_t + \alpha_t(g - H(x_t + \alpha_t(g - Hx_t))) \tag{25}$$
$$= x_t + \alpha_t(g + \alpha_t g - H + \alpha_t H^2)x_t \tag{26}$$
$$= (I - \alpha_t H + \alpha_t^2 H^2)x_t + \alpha_t(g - \alpha_t Hg) \tag{27}$$

Notice that the PC method is slower than the Euler discretization in (22).