Gradient Temporal Difference Learning

Louis de Benoist
BSc. Mathematics and Computer Science

A dissertation submitted to École Polytechnique
in partial fulfilment of the requirements for the degree of
Bachelor of Science in Mathematics and Computer Science

Centre de Mathématiques Appliquées
École Polytechnique
Palaiseau, France

Email: louis.de-benoist-de-gentissart@polytechnique.edu

Acknowledgments

I would like to thank my supervisor Professor Éric Moulines for introducing me to the topic and helping me throughout this internship at the CMAP. In addition, I also want to thank Cheikh Toure for his continuous advice, weekly meetings, and countless rereadings of this dissertation.

Contents

Chapter 1

Introduction

Reinforcement learning is one of the most exciting fields to be in right now, with the number of applications growing at an incredibly rapid pace, from beating video games to self driving cars. At its essence, reinforcement learning deals with decision making — it attempts to answer the question of how something, commonly referred to as an agent, should act in some given environment. Reinforcement learning is thus, in many ways, tied to the vague buzzword "artificial intelligence" — it is an important step towards inhibiting human-like behavior in machines.

Loosely speaking, most of reinforcement learning comes down to either finding or evaluating a policy, which is just a way of behaving in the environment. As humans, this mapping of states to actions is quite natural to think about. If we take the example of chess, a policy could be a playing strategy which takes a state, the position of all the pieces on the board, and assigns an action to it, such as moving the queen two positions forward.

Upon exiting a state, the agent observes some reward. Again, this concept models the behavior that we, ourselves, experience in the real world. For example, if we use happiness to quantify reward, putting our hand in boiling water, and thus exiting the prior state of having a comfortably cold hand, will yield some pain and, thus, a reward (in this case, negative).

In this paper, we will focus on something called policy evaluation, which boils down to evaluating a policy. Given a policy, our goal is to find its associated value function, which assigns the expected reward, or ”value”, that we expect to obtain from a given state onward provided we keep following the policy (the new states are derived from the actions prescribed by the policy that we are evaluating).

Variants of temporal difference (TD) learning, introduced by Richard Sutton in the 1980’s, are some of the most robust and used policy evaluation algorithms, especially in the on-policy setting, when we follow the same policy that we are evaluating. However, off-policy, when we follow a policy different than the one that we are trying to evaluate, convergence of TD learning algorithms is not guaranteed in some cases. This has led to the introduction of a new class of policy evaluation algorithms, gradient temporal difference learning (GTD), which guarantees convergence in the off-policy setting.

The main objective of this dissertation is to motivate and study the convergence of this family of gradient temporal difference learning algorithms and to explore through numerical simulations whether or not they are useful for concrete applications.

Chapter 2

Background

2.1 What is Reinforcement Learning?

The reinforcement learning problem deals with an agent receiving some reward signal by interacting with an environment. The agent follows some policy, a way of behaving, and tries to act so as to maximize his total reward. Policy evaluation is concerned with estimating the value of a given policy, whereas control deals with finding the optimal policy. In this paper, we'll focus on policy evaluation. [7]

2.1.1 Problem Formulation

Let $\mathbf{S}$ and $\mathbf{A}$ be finite sets and let $(S_t)_{t\in\mathbb{N}} \in \mathbf{S}^N$ be a sequence of random variables with the Markov property, $\mathbb{P}[S_{t+1} = s \mid S_t, \cdots, S_1] = \mathbb{P}[S_{t+1} = s \mid S_t]$.

At any given time $t \in \mathbb{N}$, the agent is in some state $S_t \in \mathbf{S}$ and must choose some action $A_t \in \mathbf{A}$ according to some stochastic policy $\pi : \mathbf{S} \to \mathbf{A}$. The agent then receives some scalar reward, $R_{t+1} \in \mathbb{R}$ and ends up in state S_{t+1}. The *total discounted reward* that an agent receives from time t is denoted G_t

and is written

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.1}$$

where $\gamma \in [0, 1]$ is the discount factor. Intuitively, the smaller γ is, the more we value immediate reward. Each state has a value, which is assigned by the *state-value function*, $V^\pi : \mathbf{S} \to \mathbb{R}$ given by

$$V^\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_0 = s \right] \tag{2.2}$$

To make it more condensed, we can define $V^\pi \in \mathbb{R}^{Card(S)}$ to be the vector containing values of $V^\pi(s)$ for every state $s \in \mathbf{S}$. Likewise, $R^\pi \in \mathbb{R}^{Card(S))}$ is defined as

$$R^\pi := \left(\mathbb{E}[R_{t+1} \mid S_t = s]\right)_{s \in \mathbf{S}} \tag{2.3}$$

If we use P^π to denote the state transition matrix, we can then define the *Bellman Operator* T^π

$$T^\pi V^\pi := R^\pi + \gamma P^\pi V^\pi \tag{2.4}$$

which any value function must obey. In a similar way, we can also define the *action-value function*, $Q^\pi : \mathbf{S} \times \mathbf{A} \to \mathbb{R}$,

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_0 = s,\ A_0 = a \right] \tag{2.5}$$

which is mainly used for control. However, for our purposes, we will exclusively work with the state-value function, V^π.

2.2 On-Policy Evaluation

Policy evaluation is concerned with estimating the value function for a given policy. Temporal Difference Learning (TD) algorithms, ever since their introduction in the 1980's by Richard Sutton, have become ubiquitous for policy evaluation [8] [9]. Unlike Monte Carlo, which is well known for averaging out experiences over multiple episodes, TD uses bootstrapping to learn from incomplete episodes, making it much faster and useful in concrete applications. We'll begin by considering on-policy evaluation, meaning that the exploratory policy π is the policy that we are evaluating.

2.2.1 Monte Carlo

Monte Carlo (MC) is perhaps one of the most straightforward ways to approach policy evaluation. We will introduce Monte Carlo by deriving an alternative formulation of the mean.

Consider $X_1, \cdots, X_n$ iid samples from some distribution with cdf F. Suppose that we wish to estimate $\mathbb{E}[X]$ for some $X \sim F$. A natural estimator is the sample mean, $\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^n X_i$. By doing some rewriting, we can find an incremental update to the mean [9]:

$$\begin{aligned}\hat{\mu}_n &= \frac{1}{n}\sum_{i=1}^n X_i = \frac{1}{n}(X_n + \sum_{i=1}^{n-1} X_i) = \frac{1}{n}(X_n + (n-1)\hat{\mu}_{n-1}) \\ &= \hat{\mu}_{n-1} + \frac{1}{n}(X_n - \hat{\mu}_{n-1})\end{aligned}$$

This gives us an incremental update, one that depends on the previous estimate and the new sample.

Monte Carlo policy evaluation is essentially a rewriting of the above equation with value functions. After each episode (some finite sequence of states $S_1, \cdots, S_N$), the value of a given state is updated to reflect the actual gains that were observed through the rest of the episode.

Let $\mathcal{E}^k := (S_1^k, \cdots, S_n^k)$ be the sequence of states visited in the k^{th} episode and let $N_k : \mathbf{S} \to \mathbb{N}$ be defined as

$$N_k(s) = \sum_{i=1}^k \mathbb{1}_{s \in \mathcal{E}^i} \tag{2.6}$$

i.e. the total number of updates to the value of s among all episodes. Now, suppose that $s \in \mathcal{E}^k$. Let

$$t^\star := \min \left\{ t \in \{1, \cdots, n\} \mid S_t^k = s \right\} \tag{2.7}$$

be the first time that state s is visited. Then $G^k(s)$ is defined as

$$G^k(s) := G_{t^\star} = \sum_{i=0}^{n} \gamma^i R_{t^\star+i+1} \tag{2.8}$$

More intuitively, upon having exited state s for the first time in episode k, $G^k(s)$ is the total reward that we observe from then on until the end of the episode. Knowing this, at the end of the k^{th} episode, we can formally write the Monte Carlo update to the value function estimator $\hat{V}^\pi$ at some state $s \in \mathbf{S}$ as

$$\hat{V}^\pi(s) \leftarrow \hat{V}^\pi(s) + \frac{1}{N_k(s)} \left(G^k(s) - \hat{V}^\pi(s) \right) \mathbb{1}_{s \in \mathcal{E}_k} \tag{2.9}$$

Here, $G^k(s) - \hat{V}_k^\pi(s)$ can be viewed as a measure of the error between the value that we observed and the value that we expected. Typically, what is done is that $1/N_k(s)$ is replaced by some step size α. Thus, the full Monte Carlo policy evaluation update is given by

$$\hat{V}^\pi(s) \leftarrow \hat{V}^\pi(s) + \alpha \left(G^k(s) - \hat{V}^\pi(s) \right) \mathbb{1}_{s \in \mathcal{E}_k} \tag{2.10}$$

2.2.2 TD(0)

One important shortcoming of Monte Carlo approximation is that it waits until the end of each episode to update the value function. In many cases, however, we need to learn from incomplete episodes (it is possible, for instance, that a given episode never stops). Temporal difference learning addresses this shortcoming by bootstrapping. The most basic, one-step lookahead, variant of TD is called TD(0).

To derive TD(0), we will start from Monte Carlo but, instead, update the value function without waiting until the end of the episode [9]. The only issue is that $G^k(s)$ can only be obtained once the k^{th} episode is complete, which is where the bootstrapping comes in. Suppose that at time t, we are in state S_t. From the Bellman equation,

$$\begin{aligned} G^k(S_t) &= R_{t+1} + \gamma G^k(S_{t+1}) \\ &\approx R_{t+1} + \gamma \hat{V}^\pi(S_{t+1}) \end{aligned}$$

Now that we have an approximation for $G^k(S_t)$, we can substitute it into the Monte Carlo update (for the k^{th} episode) and obtain the TD(0) algorithm, which we update at every time step t

$$\hat{V}^\pi(s) \leftarrow \hat{V}^\pi(s) + \alpha_t(R_{t+1} + \gamma \hat{V}^\pi(S_{t+1}) - \hat{V}^\pi(S_t))\mathbb{1}_{\{s=S_t\}} \tag{2.11}$$

where $(\alpha_t)_{t\in\mathbb{N}}$ is a sequence of step sizes satisfying $\sum_{t=1}^{+\infty} \alpha_t = +\infty$ and $\sum_{t=1}^{+\infty} \alpha_t^2 < +\infty$. The term $\delta_t := R_{t+1} + \gamma \hat{V}^\pi(S_{t+1}) - \hat{V}^\pi(S_t)$ is often called the TD-error and represents the difference between the TD-target, $R_{t+1} + \gamma \hat{V}_t^\pi(S_{t+1})$ and the observed value $\hat{V}^\pi(S_t)$. It is common to write the TD algorithm more compactly as

$$\hat{V}^\pi(s) \leftarrow \hat{V}^\pi(s) + \alpha_t \delta_t \mathbb{1}_{\{s=S_t\}} \tag{2.12}$$

Just like in Monte Carlo, we are updating the value function in the direction of the error, this time in the direction of the TD error. The key difference between TD and Monte Carlo to note is that TD updates the value function at every single time step, whereas Monte Carlo waits until the end of the episode.

2.2.3 State-Value Function Approximation

Up to now, we have been working with a tabular representation of these algorithms. We were updating the value function for each individual state, one at a time. In practice, this is unfeasible, especially in cases where the number of states is quite large. This is why we instead consider functional approximations of the value function, parametrized by some weight vector.

As in typical policy learning, we are still evaluating some policy, i.e trying to learn some objective function

$$V^\pi : \mathbf{S} \to \mathbb{R}$$

assigning a value for each state $s \in \mathbf{S}$. The function V_θ is parametrized by $\theta \in \mathbb{R}^n$, the parameter which we are interested in learning. We are looking for

$$V_\theta^\pi(s) \approx V^\pi(s) = \mathbb{E}_\pi[G_t \mid S_0 = s] \tag{2.13}$$

a differentiable function. Popular approximations include linear functions, i.e $V_\theta^\pi(s) = \langle \theta, \phi_s \rangle$ (where $\langle \cdot, \cdot \rangle$ denotes the usual Euclidean inner product) and nonlinear functions, such as neural networks.

2.2.4 TD(0) with State-Value Function Approximation

Deriving TD(0) with function approximation can be initially be viewed as performing a simple gradient descent (except it is not exactly, as we will see later). We first have to establish the objective function to be minimized. In the case of TD(0), we are looking to minimize the mean squared error of the value function approximation V_θ^π. Our goal is to find $\theta^\star$ such that

$$\theta^\star = \min_{\theta \in \mathbb{R}^n} \mathbb{E}_\pi \left\{ (V^\pi(S_t) - V_\theta^\pi(S_t))^2 \right\} \tag{2.14}$$

Doing gradient descent, we obtain the following update rule, where α_t is the learning rate update at time t,

$$\begin{aligned}\theta_{t+1} &= \theta_t - \frac{1}{2}\alpha_t \nabla \left[\mathbb{E}_\pi \left\{ (V^\pi(S_t))^2 - V_{\theta_t}^\pi(S_t) \right\} \right] \\ &= \theta_t + \alpha_t \mathbb{E}_\pi \left\{ (V^\pi(S_t) - V_{\theta_t}^\pi(S_t)) \nabla V_{\theta_t}^\pi(S_t) \right\}\end{aligned}$$

However, we don't know the value of $V^\pi(S_t)$ (these algorithms would be pointless if we already knew the true value function!). We thus need to bootstrap. In TD(0), what is done is that the Bellman equation is used to approximate $V^\pi(S_t)$ as

$$V^\pi(S_t) \approx \mathbb{E}_\pi[R_{t+1} + \gamma V^\pi_{\theta_t}(S_t)]$$

Thus, we obtain the new update rule,

$$\begin{aligned}
\theta_{t+1} &= \theta_t + \alpha_t \mathbb{E}_\pi \left\{ (\mathbb{E}_\pi[R_{t+1} + \gamma V^\pi_{\theta_t}(S_t)] - V^\pi_{\theta_t}(S_t)) \nabla V^\pi_{\theta_t}(S_t) \right\} \\
&= \theta_t + \alpha_t \mathbb{E}_\pi \left\{ (R_{t+1} + \gamma V^\pi_{\theta_t}(S_t) - V^\pi_{\theta_t}(S_t)) \nabla V^\pi_{\theta_t}(S_t) \right\} \\
&= \theta_t + \alpha_t \mathbb{E}_\pi \left\{ \delta_t \nabla V^\pi_{\theta_t}(S_t) \right\}
\end{aligned}$$

where $\delta_t := \delta_t(\theta_t) = R_{t+1} + \gamma V^\pi_{\theta_t}(S_{t+1}) - V^\pi_{\theta_t}(S_t)$ is the TD error. As in typical stochastic gradient descent, we sample and obtain the function approximation TD(0) algorithm [7]:

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \nabla V^\pi_{\theta_t}(S_t) \tag{2.15}$$

When it exists, the TD(0)-solution, also called the TD(0)-fixpoint, is $\theta^\star$ such that

$$\mathbb{E}\left\{ \delta_t \nabla V^\pi_{\theta^\star}(S_t) \right\} = 0 \tag{2.16}$$

TD(0) with Linear Approximation

Let $V^\pi_\theta(s) = \langle \theta, \phi_s \rangle = \theta^T \phi_s$ be the value function approximation, with $\phi(s)$ some vector of features characterizing state s. Then at some time t,

$$\nabla V^\pi_{\theta_t}(S_t) = \nabla \langle \theta, \phi(S_t) \rangle = \phi(S_t) \tag{2.17}$$

Letting $\phi_t := \phi(S_t)$, the TD(0) algorithm with linear function approximation naturally follows [8]

$$\delta_t = R_{t+1} + \gamma \theta_t^T \phi_{t+1} - \theta_t^T \phi_t \tag{2.18}$$

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t \phi_t \tag{2.19}$$

In this special case, the TD(0) solution is $\theta^\star$ satisfying

$$\mathbb{E}\left\{\delta_t \nabla V^\pi_{\theta^\star}(S_t)\right\} = \mathbb{E}\left\{\delta_t \phi_t\right\} = 0 \tag{2.20}$$

In stochastic approximation theory [7], this is typically rewritten as finding $\theta^\star$ such that

$$A\theta^\star = b \tag{2.21}$$

where $A := \mathbb{E}[\phi_t(\phi_t - \gamma\phi_{t+1})^T]$ and $b := \mathbb{E}[R_{t+1}\phi_t]$. This solution, provided it exists, will always satisfy

$$V_{\theta^\star} = \Pi(TV_{\theta^\star}) \tag{2.22}$$

where Π is a projection operator, which projects vectors v onto the nearest representable value function [10].

$$\Pi v = V_\theta \text{ where } \theta = \arg\min_\theta \|V_\theta - v\|_D^2 \tag{2.23}$$

where $D = \text{diag}[d^\pi(s_1), \cdots, d^\pi(s_n)]$ is a matrix containing the entries of the stationary distribution, d^π, weighing each error according to its probability [5]. The reason that this is done is because the Bellman operator follows the underlying state dynamics of the Markov chain, meaning that TV_θ will usually not be representable as V_θ for any θ.

In the linear case, we can write $V_\theta = \Phi\theta$ for some matrix Φ whose rows are the $\phi_s := \phi(s)$. In this special scenario, we can write Π independently of θ as

$$\Pi = \Phi\left(\Phi^\top D \Phi\right)^{-1}\Phi^\top D \tag{2.24}$$

Fig.1 gives a geometric interpretation of the projection and Bellman operators.

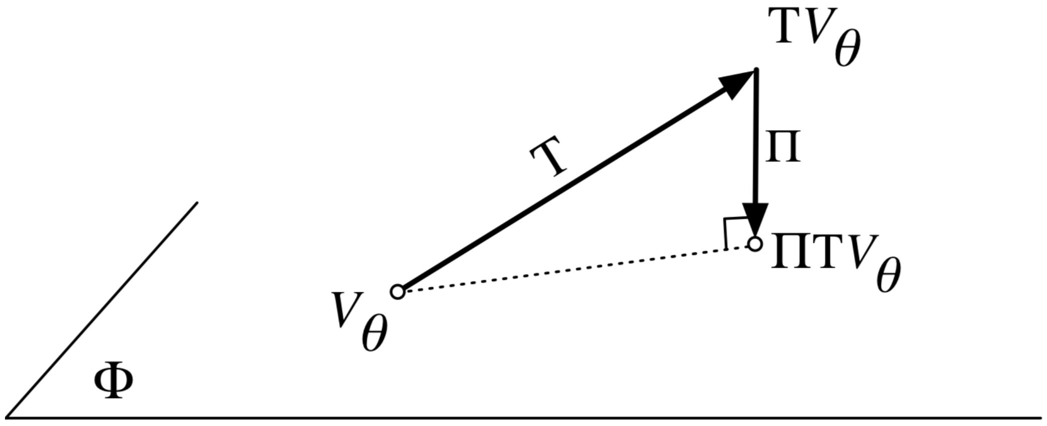


Fig. 1: Geometric relationship between the approximate value functions as well as the effect of the Bellman and projection operators on value functions. The TD(0) fixpoint satisfies $V_\theta = \Pi T V_\theta$ [7]

2.2.5 Link Between Tabular and Linear Approximation TD(0)

Although it might seem like the two versions of TD(0) that we have seen are completely different, they are in fact very much related. Indeed, the tabular version of TD(0) is a special case of TD(0) with linear function representation, as we will now show.

Let $\mathcal{S} \in \mathbb{R}^{Card(\mathbf{S})}$ be a vector containing every element of $\mathbf{S}$. We will define the feature vector to be the size of the state space, taking the value of 1 in the entry corresponding to the current state and 0 everywhere else. Formally,

$$\mathbb{R}^{Card(\mathbf{S})} \ni \phi(S_t) := (\mathbb{1}_{S_t=\mathcal{S}^{(i)}})_{i=1}^{Card(\mathbf{S})}$$

where $x^{(i)}$ denotes the i^{th} entry in some vector $x \in \mathbb{R}^n$. Then, this implies that $\theta \in \mathbb{R}^{Card(\mathbf{S})}$ and the update rule becomes

$$\theta_{t+1} = \theta_t + \alpha_t \delta_t (\mathbb{1}_{S_t=\mathcal{S}^{(i)}})_{i=1}^{Card(\mathbf{S})}$$

and, therefore,

$$\begin{cases} \theta_{t+1}^{(i)} = \theta_t^{(i)} + \alpha_t \delta_t & \text{if } S_t = i \\ \\ \theta_{t+1}^{(i)} = \theta_t^{(i)} & \text{otherwise} \end{cases}$$

Then we see that $\theta_t^{(i)} = V^\pi(S_t)$. Thus, the vector to which θ converges contains the values of the value function for every single state.

2.2.6 TD(0) Pseudocode

Algorithm 1 Tabular TD(0)

```
1:  Input: $\pi$, the policy to be evaluated
2:  $V(s) = 0 \quad \forall s \in \mathbf{S}$
3:  repeat for each episode:
4:      Initialize S, the starting state
5:      repeat at each time step:
6:          $A \leftarrow$ action given by $\pi(S)$
7:          Take action $A$, observe reward
8:          $R$, and end up in state $S'$
9:          $\delta \leftarrow R + \gamma V(S') - V(S)$
10:         $V(S) \leftarrow V(S) + \alpha\delta$
11:         $S \leftarrow S'$
12:     until S is terminal
13: until the last episode
```

Algorithm 2 Linear TD(0)

```
1:  Input: $\pi$, the policy to be evaluated
2:  Initialize $\theta$
3:  repeat for each episode
4:      Initialize S, the starting state
5:      repeat at each time step:
6:          $A \leftarrow$ action given by $\pi(S)$
7:          Take action $A$, observe reward
8:          $R$, and end up in state $S'$
9:          $\delta \leftarrow R + \gamma\theta^T\phi(S') - \theta^T\phi(S)$
10:         $\theta \leftarrow \theta + \alpha\delta\phi(S)$
11:         $S \leftarrow S'$
12:     until S is terminal
13: until the last episode
```

2.3 Off-Policy Evaluation

Reinforcement learning algorithms are very frequently run off-policy, which means that we are following some exploratory policy μ but want to evaluate another policy π. This is in practice very useful since it allows learning from others agents' experience, for example. The main question becomes: how can one estimate the expectation of a distribution given some other distribution?

Suppose that P (resp. Q) is a distribution with pdf p (resp. q). Then,

$$\begin{aligned}\mathbb{E}_{X\sim P}[f(X)] &= \sum_x f(x)p(x) = \sum_x f(x)\frac{q(x)}{q(x)}p(x) \\ &= \sum_x \left(\frac{p(x)}{q(x)}f(x)\right)q(x) = \mathbb{E}_{X\sim Q}\left[\frac{p(X)}{q(X)}f(X)\right]\end{aligned}$$

Off-policy TD(0) is given by

$$\hat{V}^\pi(s) \leftarrow \hat{V}^\pi(s) + \alpha_t\left[\frac{\pi(a|s)}{\mu(a|s)}(R_{t+1} + \gamma\hat{V}^\pi(S_{t+1})) - \hat{V}^\pi(s)\right]\mathbb{1}_{\{s=S_t\}} \tag{2.25}$$

Chapter 3

Gradient Temporal Difference Learning

3.1 Why do we need GTD methods?

Gradient temporal difference (GTD) learning methods were introduced in 2009 by Richard Sutton to address some notable shortcomings of TD. Since TD does not follow the gradient of an objective function, it is possible for it to diverge in the following cases:

1. On-policy with non-linear approximations
2. Off-policy with linear and non-linear approximations

GTD algorithms, on the other hand, will converge in all of these scenarios, making them more robust.

3.1.1 Baird's Star

Baird's Star is the most famous example demonstrating the divergence of TD in the off-policy scenario [1]. This Markov Decision Process (which can be drawn as a star, hence the name) has a fixed number of states (here, we will assume 7) and two possible actions to choose from: dotted or solid line, as can be seen in Fig. 2.

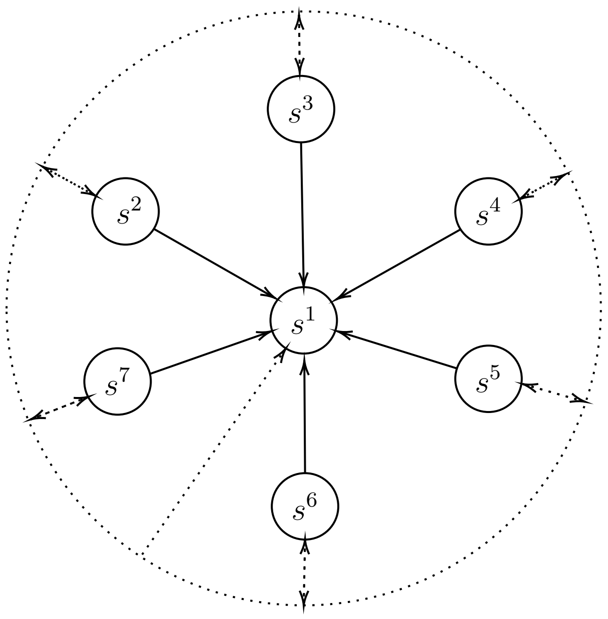


Fig. 2: Baird's Star MDP with 7 states

Choosing the dotted action will yield one of the 7 states with equal probability. Meanwhile, the solid action will take you to state s^1, the terminal state. There is 0 reward from one state to the next, meaning that the value of each state is always 0.

To observe the divergence of TD, we need to choose two different policies: an exploratory policy π_B, which we will use to navigate the environment, and the policy that we wish to evaluate, π_G. The policies are defined as follows:

$$\pi_B(\cdot|s) = \begin{cases} \frac{1}{7}, & \text{for - - -} \\ \frac{6}{7}, & \text{for —} \end{cases}$$

$$\pi_G(\cdot|s^i) = \begin{cases} 1, & \text{for - - -} \\ 0, & \text{for —} \end{cases}$$

Furthermore, the feature vector associated to state s^i is defined as

$$\phi(s^i) = \begin{cases} e_1 + 2e_7, & \text{if i = 1} \\ 2e_i + (0\ 0\ 0\ 0\ 0\ 0\ 0\ 1)^T, & \text{otherwise} \end{cases}$$

where e_i denotes the i^{th} unit vector. TD with linear approximation will diverge in this scenario with the initialization $\theta_0 = (1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 10\ \ 1)^T$ [5]. For some numerical simulations comparing different methods for various MDPs, including Baird's star, refer to Chapter 6.

3.2 Introduction to GTD

GTD methods were first introduced by Sutton in 2009 to address some of the aforementioned shortcomings of TD. In this paper, we will study, in detail, GTD methods through the lens of the GTD2 algorithm, the successor to the original GTD algorithm (which was very slow in convergence compared to conventional linear TD).

GTD2 is derived using the mean-square projected Bellman error. As a result it achieves much faster convergence than GTD, although still slower than conventional linear TD. However, the main benefit is that it will always converge in both on-policy and off-policy settings for linear approximations.

3.3 Objective Function

When deriving the functional approximation temporal difference learning algorithms, recall that we chose to minimize the Mean Squared Error (MSE). However, this was an arbitrary choice; indeed, we could have picked a different objective function, some function that we wish to minimize with respect to θ.

Mean Square Bellman Error (MSBE): The most intuitive way of measuring how much V_θ fits the Bellman equation is the Mean Square Bellman error (MSBE), given by

$$\text{MSBE}(\theta) := \mathbb{E}\left[(V_\theta - TV_\theta)^2\right] = \sum_{i=1}^{n} d_s(V_\theta^{(i)} - TV_\theta^{(i)})^2 = ||V_\theta - TV_\theta||_D^2 \quad (3.1)$$

where D is a diagonal matrix containing entries of the stationary distribution $d^\pi(s^i)$ and $||v||_D^2 := v^T Dv$ for some vector v [5].

The MSBE has been used in a lot of research but, however, most TD algorithms do not converge to its minimum. This is due to the Bellman operator following the underlying state dynamics of the Markov chain, meaning that TV_θ will usually not be representable as V_θ for any θ citeGTD2.

Mean Square Projected Bellman Error (MSPBE): The derivation of GTD2 makes use of a different objective function, the Mean Square Projected Bellman Error (MSPBE), which takes into account the projection Π, which we introduced in the previous chapter [10]. We recall that the TD solution satisfies $V_{\theta^\star} = \Pi(TV_{\theta^\star})$. The MSPBE is a measure of the deviation from the TD solution, which we will try to minimize.

$$\text{MSPBE}(\theta) = ||V_\theta - \Pi(TV_\theta)||_D^2 \quad (3.2)$$

In Fig. 3, we can see the geometric relationship between the roots of the MSBE and the MSPBE objective functions. The key difference between the two is that the MSPBE takes into account Π.

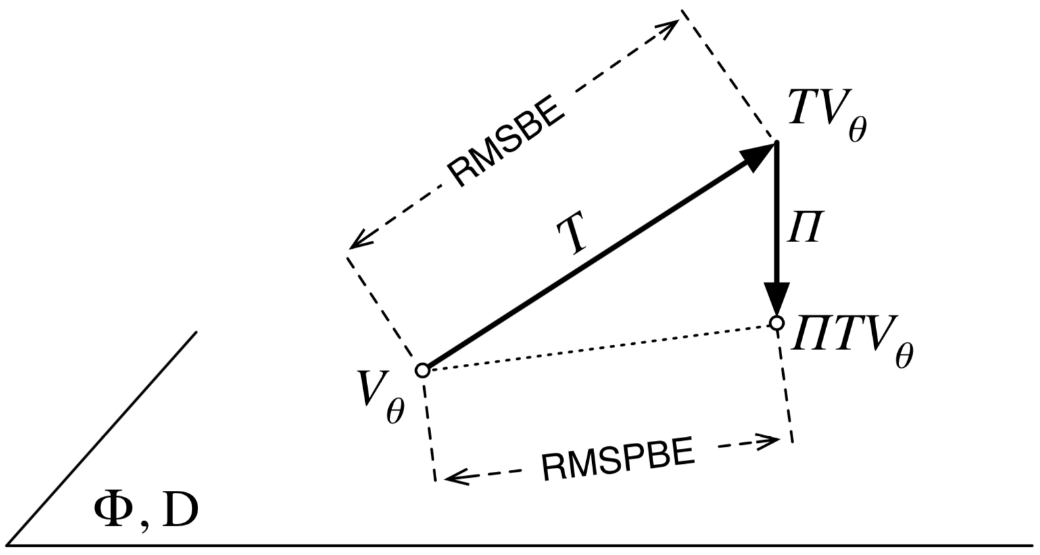


Fig. 3: Geometric relationships between the square roots of the two Bellman-error objective functions. [10]

Let us now find a simplified expression of the MSPBE as a product of expectations, a form which will allow us to derive the GTD2 algorithm. First, we need to note the following

$$\mathbb{E}[\phi\phi^\top] = \Phi^\top D\Phi \tag{3.3}$$

$$\mathbb{E}[\delta\phi] = \Phi^\top D\,(TV_\theta - V_\theta) \tag{3.4}$$

Using the above relations, we can simplify the MSPBE as

$$\begin{aligned} \text{MSPBE}(\theta) &= ||V_\theta - \Pi(TV_\theta)||_D^2 && (3.5)\\ &= ||\Pi(V_\theta - TV_\theta)||_D^2 \quad \text{since } \Pi V_\theta = V_\theta && (3.6)\\ &= [\Pi(V_\theta - TV_\theta)]^T D[\Pi(V_\theta - TV_\theta)] && (3.7)\\ &= (V_\theta - TV_\theta)^T(\Pi^T D\Pi)(V_\theta - TV_\theta) && (3.8) \end{aligned}$$

As an intermediary step, notice that

$$\Pi^\top D\Pi = \left(\Phi\left(\Phi^\top D\Phi\right)^{-1}\Phi^\top D\right)^\top D\left(\Phi\left(\Phi^\top D\Phi\right)^{-1}\Phi^\top D\right) \tag{3.9}$$

$$= D^\top\Phi\left(\Phi^\top D\Phi\right)^{-1}\Phi^\top D\Phi\left(\Phi^\top D\Phi\right)^{-1}\Phi^\top D \tag{3.10}$$

$$= D^\top\Phi\left(\Phi^\top D\Phi\right)^{-1}\Phi^\top D \tag{3.11}$$

Now, by replacing $(\Pi^T D\Pi)$ and simplifying, we can obtain a simpler version of the MSPBE as follows:

$$\text{MSPBE}(\theta) = (V_\theta - TV_\theta)^T D^\top\Phi\left(\Phi^\top D\Phi\right)^{-1}\Phi^\top D(V_\theta - TV_\theta) \tag{3.12}$$

$$= \left[\Phi^T D(TV_\theta - V_\theta)\right]^T\left[\Phi^\top D\Phi\right]^{-1}\left[\Phi^\top D(TV_\theta - V_\theta)\right] \tag{3.13}$$

$$= \mathbb{E}[\delta\phi]^T\,\mathbb{E}[\phi\phi^T]^{-1}\,\mathbb{E}[\delta\phi] \tag{3.14}$$

It is interesting to note that to derive the original GTD algorithm, the objective function that was used was the *norm of the expected TD update* (NEU) [11],

$$\text{NEU}(\theta) = \mathbb{E}[\delta\phi]^\top\mathbb{E}[\delta\phi] \tag{3.15}$$

We see that the MSPBE only differs from the NEU by the inclusion of the inverse of the feature covariance matrix.

3.4 Derivation of GTD2

Let us now use this new expression of the MSPBE to derive the GTD2 algorithm, i.e. we will perform a gradient descent on the MSPBE objective function. To avoid needing two independent samples, we can use a modifiable parameter $w \in \mathbb{R}^n$ to form a quasi-stationary estimate of all but one of the expectations in the gradient of the objective function

$$w \approx \mathbb{E}[\phi\phi^T]^{-1}\,\mathbb{E}[\delta\phi] \tag{3.16}$$

Using this, we obtain

$$-\frac{1}{2}\alpha\nabla\mathrm{MSPBE}(\theta) = \alpha\mathbb{E}\left[(\phi-\gamma\phi')\,\phi^\top\right]\mathbb{E}\left[\phi\phi^\top\right]^{-1}\mathbb{E}[\delta\phi] \tag{3.17}$$

$$\approx \alpha\mathbb{E}\left[(\phi-\gamma\phi')\,\phi^\top\right]w \tag{3.18}$$

which can be directly sampled. From this, we obtain the GTD2 algorithm:

$$\theta_{k+1} = \theta_k + \alpha_k(\phi_k - \gamma\phi_k')(\phi_k^T w_k) \tag{3.19}$$

Now, we want to find an iterative update for w. First, we realize that

$$w \approx \mathbb{E}[\phi\phi^T]^{-1}\,\mathbb{E}[\delta\phi] = (\Phi^T D\Phi)^{-1}\Phi^T D(TV_\theta - V_\theta)$$

We see that the right side of the above is a solution to the following least squares problem:

$$||\Phi^T w - (TV_\theta - V_\theta)||_2^2$$

which can be solved by stochastic gradient descent with the following update:

$$w_{k+1} = w_k + \beta_k\left(\delta_k - \phi_k^\top w_k\right)\phi_k \tag{3.20}$$

3.5 One-timescale and Two-timescale Stochastic Approximations

GTD2 is an instance of a two-timescale stochastic approximation (SA) algorithm. In general, two-timesecale SA algorithms can be written as

$$\theta_{k+1} = \theta_k + \alpha_k(h_1(\theta_k, w_k) + M_{k+1}^{(1)}) \tag{3.21}$$

$$w_{k+1} = w_k + \beta_k(h_2(\theta_k, w_k) + M_{k+1}^{(2)}) \tag{3.22}$$

where α_k and β_k are stepsizes and $M_{k+1}^{(i)} \in \mathbb{R}^d$ is some noise, usually Martingale or Markovian [4].

An algorithm is said to be run in one-timescale mode if $\alpha_k/\beta_k = K$ for some constant $K \in \mathbb{R}$. In two-timescale mode, however, the assumption is that $\alpha_k/\beta_k \to 0$. According to Dalal, one-timescale mode with $\alpha_k = \beta_k \approx \frac{1}{k}$ will usually yield better results than two-timescale, a claim which we will verify in Chapter 6 [4].

3.6 GTD2 Pseudocode

Algorithm 3 GTD2(0)

```
 1: Input: $\pi$, the policy to be evaluated
 2: Initialize $\theta$
 3: **repeat** for each episode:
 4:     Initialize S, the starting state
 5:     **repeat** at each step in the episode:
 6:         $A \leftarrow$ action given by $\pi$ for state $S$
 7:         Take action $A$, observe reward $R$,
 8:         and end up in state $S'$
 9:         $\delta \leftarrow R + \gamma V(S') - V(S)$
10:         $\theta \leftarrow \theta + \alpha(\phi(S) - \gamma\phi(S'))\phi(S)^T w$
11:         $w \leftarrow w + \beta(\delta - \phi(S)^T w)\phi(S)$
12:         $S \leftarrow S'$
13:     **until** S is terminal
14: **until** the last episode
```

Chapter 4

Asymptotic Convergence of GTD2

Now that we have derived the GTD2 algorithm, we will rigorously prove that it is indeed asymptotically convergent. The proof itself is very heavily dependent on the so-called "ODE approach" for recursive stochastic algorithm, a tool which we will now introduce[2].

4.1 ODE Approach for Recursive Stochastic Algorithms

We consider stochastic algorithms, algorithms of the form

$$x_{k+1} = x_k + \alpha_k \left(h\left(x_k \right) + M_{k+1} \right) \tag{4.1}$$

for some $x \in \mathbb{R}^d$, $h : \mathbb{R}^d \to \mathbb{R}^d$ differentiable, $(a_k)_{k \geq 0}$ a positive step-size sequence, and $(M_k)_{k \geq 0}$ a noise sequence. We suppose that the above equation is a discretization of

$$\frac{dx}{dt} = h(x(t))$$

This ODE approach consists in deducing the convergence of the algorithm by studying at the convergence behavior of the ODE.

ODE Theorem (Theorem 2.2 of Borkar and Meyn). [2] Consider Equation (4.1) and suppose that the following conditions hold:

- (i) $\sum_k \alpha_k = +\infty, \sum_k \alpha_k^2 < \infty$
- (ii) The function h is Lipschitz and $h_\infty(x) = \lim_{c\to\infty} \frac{h(cx)}{c}$ is well defined for any $x \in \mathbb{R}^d$.
- (iii) The sequence $(M_k)_{k\geq 0}$ is a martingale difference sequence with respect to $\mathcal{F}_k := \sigma(x_0, M_1, \cdots, M_k)$, i.e. $\mathbb{E}[M_{k+1} \mid \mathcal{F}_k] = 0$.
- (iv) There exists $K > 0$ such that for all $k > 0$, $\mathbb{E}\left[\|M_{k+1}\|^2 |\mathcal{F}_k\right] \leq K\left(1 + \|x_k\|^2\right)$ almost surely.
- (v) The ODE $\frac{dx}{dt} = h_\infty(x)$ has the origin as a globally asymptotically stable equilibrium
- (vi) The ODE $\frac{dx}{dt} = h(x)$ has a unique globally asymptotically stable equilibrium.

Then, as $k \to \infty$, $(x_k)_{k\geq 0}$ converges with probability one to the unique globally asymptotically stable equilibrium of the ODE, $\frac{dx}{dt} = h(x(t))$.

4.2 Proof of Convergence

Theorem 1 (Convergence of GTD2). [10] Consider the GTD2 iterations (3.4) and (3.4) with step-size sequences α_k and β_k satisfying $\beta_k = \eta\alpha_k$, $\eta > 0$, $\alpha_k, \beta_k \in]0, 1]$, and $\sum_{k=0}^\infty \alpha_k = \infty, \sum_{k=0}^\infty \alpha_k^2 < \infty$. Furthermore, suppose that (ϕ_k, r_k, ϕ'_k) is an i.i.d. sequence with uniformly bounded second moments. Let

$$A = \mathbb{E}[\phi_k(\phi_k - \gamma\phi'_k)^T], \quad b = \mathbb{E}[r_k\phi_k], \quad C = \mathbb{E}[\phi_k\phi_k^T]$$

and suppose that A and C are non-singular. Then the parameter vector θ_k converges with probability one to the TD fixpoint.

Proof. We will begin by showing that the TD fixpoint can be written as the condition

$$-A\theta + b = 0 \tag{4.2}$$

First, we see that θ is the TD fixpoint if $\text{MSPBE}(\theta) = 0$. Thus,

$$\text{MSPBE}(\theta) = \mathbb{E}[\delta\phi]^T\ \mathbb{E}[\phi\phi^T]^{-1}\ \mathbb{E}[\delta\phi] = ||\mathbb{E}[\delta\phi]||^2_{E[\phi\phi^T]^{-1}} = 0$$

which implies that $\mathbb{E}[\delta\phi] = 0$. Now, we see that

$$\begin{aligned}
\mathbb{E}[\delta\phi] &= \Phi^\top D\,(TV_\theta - V_\theta) \\
&= \Phi^T D(R + \gamma\theta^T\phi' - \theta^T\phi) = \Phi^T DR + \Phi^T D(\gamma\theta^T\phi' - \theta^T\phi) \\
&= \mathbb{E}[r\phi] - \Phi^T D(\theta^T\phi - \gamma\theta^T\phi') = \mathbb{E}[r\phi] - \Phi^T D\theta^T(\phi - \gamma\phi') \\
&= \mathbb{E}[r\phi] - \Phi^T D(\phi - \gamma\phi')^T\theta = \mathbb{E}[r\phi] - \mathbb{E}[\phi(\phi - \gamma\phi')^T]\theta \\
&= b - A\theta
\end{aligned}$$

Now, we're going to rewrite the algorithm's two iterative updates (for θ_k and w_k) by combining them into one parameter vector, $\rho_k = (d_k^T, \theta_k^T)^T$, where $d_k := \frac{w_k}{\sqrt{\eta}}$. Furthermore, let $g_{k+1} = (r_k\phi_k^T, 0^T)^T$. Let's consider the following

$$\rho_{k+1} = \rho_k + \alpha_k\sqrt{\eta}(G_{k+1}\rho_k + g_{k+1})$$

where

$$G_{k+1} = \begin{pmatrix} -\sqrt{\eta}\phi_k\phi_k^\top & \phi_k\,(\gamma\phi_k' - \phi_k)^\top \\ (\phi_k - \gamma\phi_k')\,\phi_k^\top & 0 \end{pmatrix}$$

Now, let $G = \mathbb{E}[G_k]$ and $g = \mathbb{E}[g_k]$. We obtain

$$G = \begin{pmatrix} -\sqrt{\eta}C & -A \\ A^\top & 0 \end{pmatrix}, \quad g = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

Let $\rho = (d^T, \theta^T)^T$. Let's suppose that we can show that $G\rho + g = 0$. Multiplying everything out, we obtain that $A\theta = b$, i.e that θ is a fixed point. It is thus sufficient to show that $G\rho + g = 0$. To do this, we're going to use the

ODE approach, applying Theorem 2.2 of Borkar and Meyn [2]. Let's write

$$\begin{aligned}\rho_{k+1} &= \rho_k + \alpha_k\sqrt{\eta}(G\rho_k + g + (G_{k+1} - G)\rho_k + (g_{k+1} - g)) \\ &= \rho_k + \alpha'_k(h(\rho_k) + M_{k+1})\end{aligned}$$

where $\alpha'_k = \alpha_k\sqrt{\eta}$, $h(\rho) = g + G\rho$, and $M_{k+1} = (G_{k+1} - G)\rho_k + g_{k+1} - g$. Let $\mathcal{F}_k = \sigma(\rho_1, M_1, \cdots, \rho_{k-1}, M_k)$ be the sigma algebra generated by ρ_i and M_i. Let's now check that we verify the conditions for the ODE theorem.

- (i) Because of the conditions imposed on α_k and β_k, we obtain that

$$\sum_k \alpha'_k = \sqrt{\eta}\sum_k \alpha_k = \infty$$

  and

$$\sum_k \alpha'^2_k = \eta\sum_k \alpha^2_k < \infty$$

- (ii) h is Lipschitz. Indeed, take $x, y \in \mathbb{R}^{2n}$. Then,

$$\begin{aligned}||h(x) - h(y)|| = ||g + Gx - (g + Gy)|| &= ||G(x - y)|| \\ &\leq ||G||\ ||x - y||\end{aligned}$$

  Furthermore, $h_\infty(\rho)$ is well defined as

$$h_\infty(\rho) = \lim_{r\to\infty} \frac{h(r\rho)}{r} = \lim_{r\to\infty} \frac{g + Gr\rho}{r} = G\rho$$

  for every $\rho \in \mathbb{R}^{2n}$.

- (iii) $(M_k, \mathcal{F}_k)$ is a Martingale difference sequence. Indeed,

$$\begin{aligned}\mathbb{E}[M_{k+1} \mid \mathcal{F}_k] &= \mathbb{E}[(G_{k+1} - G)\rho_k + g_{k+1} - g \mid \mathcal{F}_k] \\ &= \rho_k\mathbb{E}[G_{k+1} \mid \mathcal{F}_k] + \mathbb{E}[g_{k+1} \mid \mathcal{F}_k] - G\rho_k - g \\ &= \rho_k G + g - G\rho_k - g = 0\end{aligned}$$

- (iv) There exists $c_0 > 0$ such that $\mathbb{E}[\ ||M_{k+1}||^2 \mid \mathcal{F}_k] \leq c_0(1 + ||\rho_k||^2)$

for any initial parameter ρ_1. Indeed, thanks to the triangle inequality,

$$\begin{aligned}
\mathbb{E}[\ ||M_{k+1}||^2 \mid \mathcal{F}_k] &= \mathbb{E}[\ ||(G_{k+1}-G)\rho_k + (g_{k+1}-g)||^2 \mid \mathcal{F}_k] \\
&\leq 2\mathbb{E}[\ ||(G_{k+1}-G)\rho_k||^2 \mid \mathcal{F}_k] + 2\mathbb{E}[\ ||g_{k+1}-g||^2 \mid \mathcal{F}_k] \\
&\leq 2||\rho_k||^2\mathbb{E}[\ |||G_{k+1}-G|||^2 \mid \mathcal{F}_k] + 2\mathbb{E}[\ ||g_{k+1}-g||^2 \mid \mathcal{F}_k]
\end{aligned}$$

Hence, item (iv) follows from the boundedness of the second moments of (ϕ_k, r_k, ϕ'_k).

- (v) Let's show that $\frac{d\rho}{dt} = h_\infty(\rho) = G\rho$ has the origin as a globally asymptotically stable equilibrium. To do that, we will show that $\Re(\lambda_i) < 0$ for every eigenvalue λ_i of G (condition found on slide 13-4). First, let's show that G is not singular. Using the formula for computing the determinant of a block matrix,

$$\det(G) = \det(A^T C^{-1} A) = (\det(C))^{-1}(\det(A))^2 \neq 0$$

since C and A are non-singular by assumption. Thus, all eigenvalues of G are non-zero. Now, take $\lambda \in \mathbb{C}, \lambda \neq 0$, an eigenvalue of G. Let's show that $\Re(\lambda) < 0$. Let $x \in \mathbb{R}^{2n}$ be its corresponding normalized eigenvector, i.e $||x|| = 1$. Then, if we let $x^\star$ denote its complex conjugate, it follows that $x^\star x = 1$ and, thus,

$$x^\star G x = \lambda$$

Let $x = (x_1^T, x_2^T)^T$ with $x_1, x_2 \in \mathbb{C}^n$. Multiplying through, we obtain that

$$\begin{aligned}
\lambda = x^\star G x &= \begin{pmatrix} x_1^{\star T} \\ x_2^{\star T} \end{pmatrix} \begin{pmatrix} -\sqrt{\eta}C & -A \\ A^\top & 0 \end{pmatrix} \begin{pmatrix} x_1^T \\ x_2^T \end{pmatrix} \\
&= -\eta(x_1^\star C x_1) - x_1^\star A x_2 + x_2^\star A^T x_1 \\
&= -\eta||x_1||_C^2 - x_1^\star A x_2 + x_2^\star A^T x_1
\end{aligned}$$

where $||x_1||_C^2 = x_1^\star C x_1$. Since A is real, $A^T = A^\star$ and, thus, $x_2^\star A^T x_1 =$

$(x_1^\star A x_2)^\star$. Hence,

$$\Re(\lambda) = \Re(x^\star G x) = -\eta ||x_1||_C^2 \leq 0$$

Now, we just need to check that $x_1 \neq 0$. Suppose that $x_1 = 0$. Then, $\lambda = x^\star G x = 0$, which is a contradiction. Thus, $\Re(\lambda) < 0$ and we conclude that G has the origin as a globally asymptotically stable equilibrium.

- (vi) Let's show that $\frac{d\rho}{dt} = h(\rho)$ has a unique globally asymptotically stable equilibrium. First of all, we see that

$$\frac{d\rho}{dt} = 0 \implies h(\rho) = 0 \implies g + G\rho = 0 \implies \rho = -G^{-1}g$$

Hence, $\rho^\star = -G^{-1}g$ is the equilibrium. Now, we want to show that it is an asymptotically stable equilibrium. To do that, we will find a strict Lyapunov function to show that the origin of $\frac{dx}{dt} = \frac{d\rho}{dt} = h(\rho) = h(x+\rho^\star)$ is a globally asymptotically stable equilibrium where $x := \rho - \rho^\star$. We claim that the ODE's associated strict Lyapunov function is

$$\bar{V}(x) = \frac{1}{2}(G(x+\rho^\star)+g)^T(G(x+\rho^\star)+g) = \frac{1}{2}||h(x+\rho^\star)||^2$$

Let's show that $\bar{V}$ is a Lyapunov function.

  - We will show that $\bar{V}(0) = 0$.

$$\bar{V}(0) = \frac{1}{2}||h(0+\rho^\star)||^2 = \frac{1}{2}||h(\rho^\star)||^2 = 0$$

    since $\rho^\star$ is the equilibrium.

  - We will show that $\bar{V}(x) > 0$ for all $x \neq 0$. $\bar{V}$ is quadratic and, thus, positive for any $x \neq 0$.

  - We will show that $\frac{d\bar{V}}{dt}$ is negative definite, meaning that for all t, the derivative at t, $(\bar{V} \circ x)'(t)$ is always negative. Thanks to the chain rule derivative, it follows that

$$(\bar{V} \circ x)'(t) \quad = \quad \nabla \bar{V}(x(t)) \cdot x'(t), \tag{4.3}$$

which is the scalar product between the gradient $\bar{V}(x(t))$ and the derivative $x'(t)$.

Recall that $x'(t) = h(x(t) + \rho^\star) = Gx(t) + G\rho^\star + g = Gx(t)$.

Also, $\bar{V}(x) = \frac{1}{2}||h(x + \rho^\star)||^2 = \frac{1}{2}||Gx||^2$. Hence,

$$\bar{V}(x) = \frac{1}{2}(Gx)^T Gx = \frac{1}{2}x^T G^T Gx. \tag{4.4}$$

This equation ensures that $\nabla \bar{V}(x) = G^T Gx$. Hence Equation (4.3) becomes

$$\begin{aligned}
(\bar{V} \circ x)'(t) &= G^T Gx(t) \cdot Gx(t) && (4.5)\\
&= (Gx(t))^T G^T Gx(t) && (4.6)\\
&= x(t)^T G^T G^T Gx(t). && (4.7)
\end{aligned}$$

It follows from the transposition of the right-hand side last equation (which is a real number) that:

$$(\bar{V} \circ x)'(t) \quad = \quad x(t)^T G^T GGx(t). \tag{4.8}$$

Thereby,

$$2(\bar{V} \circ x)'(t) \quad = \quad (Gx(t))^T (G + G^T) Gx(t). \tag{4.9}$$

It is then enough to show that $G + G^T$ is a negative definite matrix, to show that

$$(\bar{V} \circ x)'(t) < 0.$$

This is the case because $G + G^T = \begin{pmatrix} -2\sqrt{\eta}C & 0 \\ 0 & 0 \end{pmatrix}$ is negative definite since C is positive definite.

Then, we can apply Theorem 2.2 and obtain that $(\rho_k)_{k\geq 0}$ converges with probability one to the unique globally asymptotically stable equilibrium of the ODE. Thus, ρ_k converges to ρ and, since it is an equilibrium, we have $\frac{d\rho}{dt} = h(\rho) = g + G\rho = 0$, which proves convergence to the fixed point.

$\square$

Chapter 5

Finite Time Convergence of GTD Methods

Now that we have established the asymptotic convergence of GTD2, the questions becomes: how fast does GTD2 converge? In practice, the rate at which the algorithm converges determines its usability, making this a very important question. In recent years, the focus has been on performing a non-asymptotic analysis of the broader category of two-timescale stochastic approximation algorithms, which we introduced back in Chapter 3. In this chapter, we will lay out some of the key non-asymptotic results that have been found over time.

Finite-Sample Analysis of Proximal Gradient TD Algorithms
One of the first influential works in the non-asymptotic analysis, this paper contributed to the field by converting the objective function of GTD algorithms into a convex-concave saddle problem. This paper also included the first finite sample analysis for GTD with constant step size under the assumption that the data is i.i.d. generated, which is not very realistic.[6]

Finite Sample Analysis of the GTD Policy Evaluation Algorithms

in Markov Setting

This paper is an extension of *Finite-Sample Analysis of Proximal Gradient TD Algorithms*, which we just mentioned above. This paper's fnite-time analysis involved some important changes, notably [12] :

- Changing the i.i.d assumption to a more realistic Markovian sampling condition.
- Modifying the step-size setting
- Discussing how experience replay can speed up the Markov chain's mixing time.

A Tale of Two-Timescale Reinforcement Learning with the Tightest Finite-Time Bound

This paper obtained some important bounds for two-timescale algorithms. Under the assumption that $\alpha_n = \frac{1}{n^\alpha}$ and $\beta_n = \frac{1}{n^\beta}$ with $0 < \beta < \alpha < 1$, it is shown that $\theta_n \to \theta^\star$ and $w_n \to w^\star$ with the following rates:

$$||\theta_n - \theta^\star|| = \tilde{O}\left(\tfrac{1}{n^{\alpha/2}}\right) \tag{5.1}$$

$$||w_n - w^\star|| = \tilde{O}\left(\tfrac{1}{n^{\beta/2}}\right) \tag{5.2}$$

where $\tilde{O}$ disregards logarithmic terms. Besides this bound, which was also shown to be tight, it is also demonstrated that there exists some finite time from which the convergence rates of θ_n and w_n no longer depend on α and β. It was previously only shown that this decoupling happened asymptotically.[4]

Chapter 6

Numerical Simulations

Having studied the convergence of the GTD2 algorithm, we will now see in practice how it truly fares in comparison to Monte Carlo and TD(0). Furthermore, we will also illustrate the effects of choosing to use GTD2 in one-time scale versus two-time scale. As case studies, we will focus on Boyan chains and Baird's star.

6.1 Implementation Details

All of the implementations were done in Python 3 and can be found on the following Gitlab link:
`https://gitlab.com/cheikh_toure/bachelor-project-on-reinforcement-learning/`
`-/tree/master/Implementations`

6.2 Boyan Chain

Boyan's chain, introduced by Boyan in 2002, has become one of the most common benchmarks for reinforcement learning algorithms [3]. It is a very simple object, a chain (as the name might indicate), with a fixed number of states, transition probabilities, and rewards. There are no actions to choose from, making this a Markov Reward Process (MRP) rather than a Markov

Decision Process (MDP). For our simulation, we will consider a 7-state Boyan Chain, defined as follows.

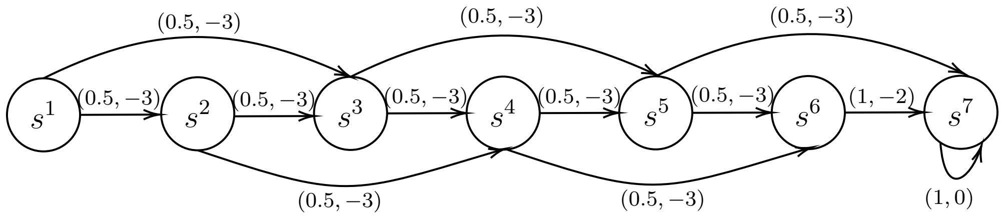


Fig. 4: Boyan chain with 7 states. The transitions are labelled (p, r) where p is the transition probability and r is the reward.

The true value function of this 7-state Boyan Chain is given by

$$V(s_i) = -12 + 2(i-1) \tag{6.1}$$

It has been previously shown that the following feature vectors are capable of exactly representing the value function,

$$\phi(s_i) = \left(\frac{7-i}{6}\, ,\ \frac{i-1}{6}\right)^T \tag{6.2}$$

Using this set of features [5], it has been shown that $\theta^\star = [-12\ 0]^T$. In this simulation, GTD2 with one-timescale had the following learning rate updates:

$$\alpha_t = \beta_t = \frac{1}{t} \tag{6.3}$$

In the two-timescale case, the following updates were used

$$\alpha_t = \frac{1}{t^2}, \quad \beta_t = \frac{1}{t} \tag{6.4}$$

Fig. 5 shows the RMSE over time of the different algorithms, all of which converged. In this specific case, Monte Carlo proved to be the fastest, followed by one-timescale GTD2 and TD, confirming Dalal's prediction that a one-timescale configuration with $\alpha_t = \beta_t \approx \frac{1}{t}$ would yield close to optimal convergence in practice, when compared to two-timescale GTD2 [4].

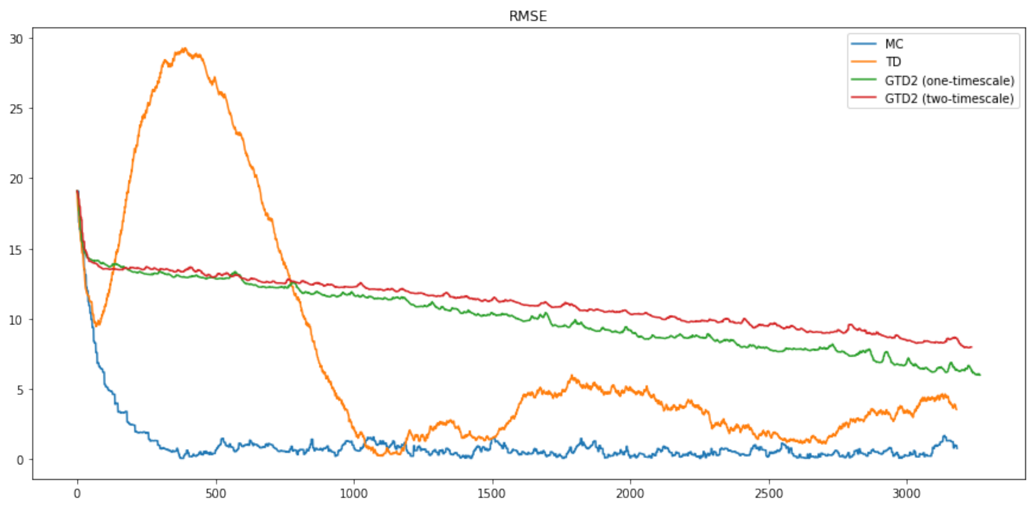


Fig. 5: Root mean square errors of Monte Carlo, TD(0), and GTD2(0) (one and two timescales)

To obtain a plot of the RMSPBE, we need to obtain the stationary distribution of the Markov chain, d^π (which we need to compute the operator Π). Recall that d^π satisfies

$$d^\pi = d^\pi P \tag{6.5}$$

meaning that d^π is invariant by P, the transition matrix. Transposing both sides, we get

$$P^T {d^\pi}^T = {d^\pi}^T$$

meaning that d^π is the transposed eigenvector associated to the eigenvalue $\lambda = 1$ of the transition matrix. Here, it is important to note that P is not the Markov Chain associated to the graphical representation of the 7-state Boyan chain, but rather, it is the Markov Chain associated to the stochastic process that will be undertaken in RL, which involves resetting the Markov chain at the end of every episode. In this case,

$$P = \begin{pmatrix} 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Diagonalizing P^T and normalizing the eigenvector associated to $\lambda = 1$, we obtain that the stationary distribution is $d^\pi = (0.192, 0.096, 0.144, 0.120, 0.132, 0.126, 0.192)$. Thus,

$$\Pi = \Phi \left(\Phi^\top D \Phi\right)^{-1} \Phi^\top D$$

with $D = \text{diag}[1,\ 0.5,\ 0.75,\ 0.625,\ 0.6875,\ 0.65625,\ 1]$ and

$$\Phi^T = \begin{pmatrix} 1 & 5/6 & 4/6 & 3/6 & 2/6 & 1/6 & 0 \\ 0 & 1/6 & 2/6 & 3/6 & 4/6 & 5/6 & 1 \end{pmatrix}$$

Thus,

$$\Pi = \begin{pmatrix} 0.58 & 0.23 & 0.24 & 0.12 & 0.05 & -0.04 & -0.18 \\ 0.45 & 0.18 & 0.21 & 0.12 & 0.08 & 0.02 & -0.06 \\ 0.32 & 0.14 & 0.18 & 0.12 & 0.10 & 0.07 & 0.06 \\ 0.20 & 0.10 & 0.15 & 0.12 & 0.13 & 0.12 & 0.19 \\ 0.07 & 0.06 & 0.11 & 0.12 & 0.16 & 0.18 & 0.31 \\ -0.06 & 0.01 & 0.08 & 0.12 & 0.18 & 0.23 & 0.43 \\ -0.18 & -0.03 & 0.05 & 0.12 & 0.21 & 0.28 & 0.55 \end{pmatrix} \tag{6.6}$$

which we can then use to obtain a plot of the RMSPBE.

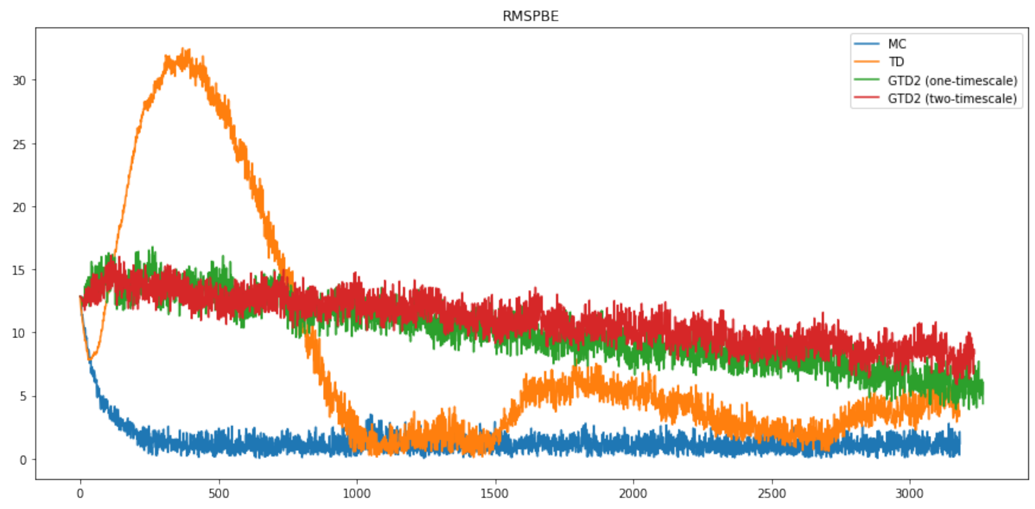
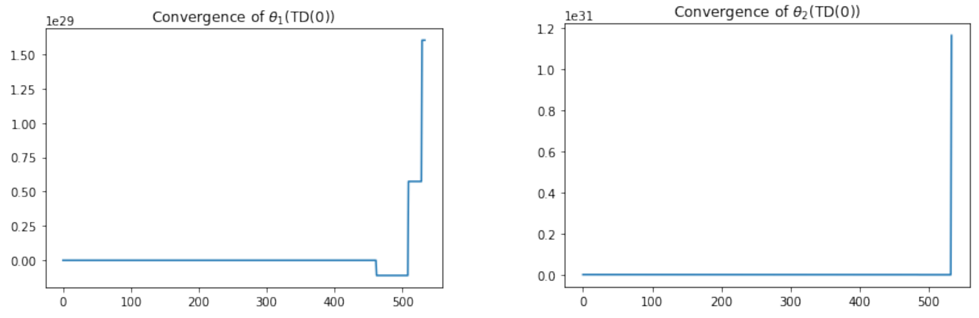


Fig. 6: Root mean square projected Bellman errors of Monte Carlo, TD(0), and GTD2(0) (one and two timescales)

6.3 Baird's Star

Back in chapter 3, we introduced Baird's star, one of the most famous examples for which TD diverges when off-policy. As expected, when using TD(0), $||\theta_t|| \to \infty$ as $t \to \infty$. Fig. 6 shows the divergence of some of the components of θ_t.



(a) Divergence of $\theta^{(1)}$ over time

(b) Divergence of $\theta^{(2)}$

Fig. 6: Illustration of the divergence of the entries of θ with TD(0) over time

However, also unsurprisingly, GTD2 converged in a neighborhood of the optimal value function, as can be seen in Fig. 7. The plot showcases the differences between one-timescale and two-timesecale simulations with GTD2. The hyperparameters were tuned so as to obtain clean and quick convergence. In the one-timescale plot, $\alpha_k = \beta_k = \frac{1}{(k+3)^{7/2}}$ and in the two-timescale case, $\alpha_k = \frac{1}{(k+3)^5}$ and $\beta = \frac{1}{(k+2)^2}$

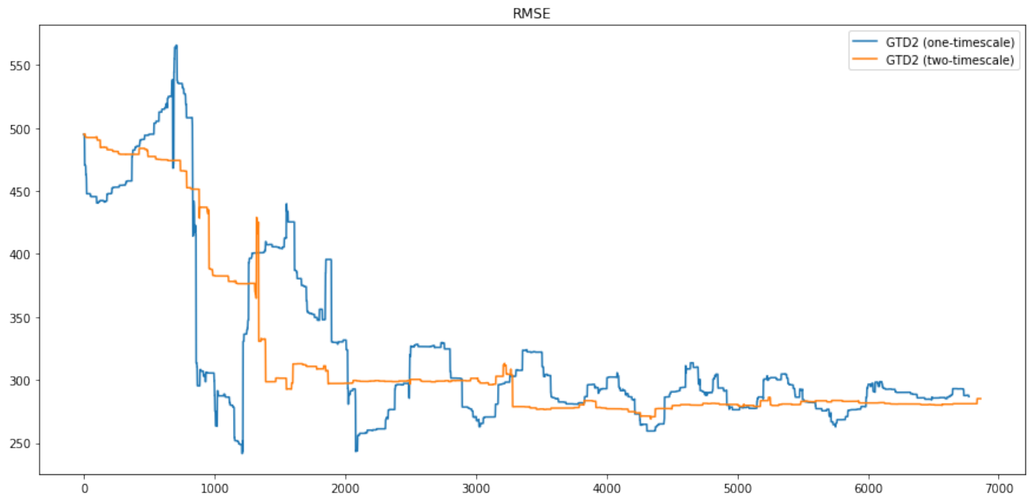


Fig. 7: Root mean square errors of GTD2(0) (one and two timescales) for Baird's star with 7 states

Bibliography

[1] Leemon Baird. “Residual Algorithms: Reinforcement Learning with Function Approximation”. In: *In Proceedings of the Twelfth International Conference on Machine Learning.* Morgan Kaufmann, 1995, pp. 30–37.

[2] V. S. Borkar and S. P. Meyn. “The O.D. E. Method for Convergence of Stochastic Approximation and Reinforcement Learning”. In: *SIAM J. Control Optim.* 38.2 (Jan. 2000), pp. 447–469. ISSN: 0363-0129. DOI: 10.1137/S0363012997331639. URL: https://doi.org/10.1137/S0363012997331639.

[3] Justin A. Boyan. “Technical Update: Least-Squares Temporal Difference Learning”. In: *Mach. Learn.* 49.2–3 (Nov. 2002), pp. 233–246. ISSN: 0885-6125. DOI: 10.1023/A:1017936530646. URL: https://doi.org/10.1023/A:1017936530646.

[4] Gal Dalal, Balazs Szorenyi, and Gugan Thoppe. *A Tale of Two-Timescale Reinforcement Learning with the Tightest Finite-Time Bound.* 2019. arXiv: 1911.09157 [cs.LG].

[5] Christoph Dann, Gerhard Neumann, and Jan Peters. “Policy Evaluation with Temporal Differences: A Survey and Comparison”. In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 809–883. ISSN: 1532-4435.

[6] Bo Liu et al. “Finite-Sample Analysis of Proximal Gradient TD Algorithms”. In: *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence.* UAI’15. Amsterdam, Netherlands: AUAI Press, 2015, pp. 504–513. ISBN: 9780996643108.

[7] Hamid Reza Maei. "Gradient Temporal-Difference Learning Algorithms". PhD thesis. CAN, 2011. ISBN: 9780494894552.

[8] Richard S. Sutton. "Learning to Predict by the Methods of Temporal Differences". In: *Mach. Learn.* 3.1 (Aug. 1988), pp. 9–44. ISSN: 0885-6125. DOI: `10.1023/A:1022633531479`. URL: `https://doi.org/10.1023/A:1022633531479`.

[9] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: `http://incompleteideas.net/book/the-book-2nd.html`.

[10] Richard S. Sutton et al. "Fast Gradient-Descent Methods for Temporal-Difference Learning with Linear Function Approximation". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 993–1000. ISBN: 9781605585161. DOI: `10.1145/1553374.1553501`. URL: `https://doi.org/10.1145/1553374.1553501`.

[11] Richard S Sutton, Hamid R. Maei, and Csaba Szepesvári. "A Convergent O(n) Temporal-difference Algorithm for Off-policy Learning with Linear Function Approximation". In: *Advances in Neural Information Processing Systems 21*. Ed. by D. Koller et al. Curran Associates, Inc., 2009, pp. 1609–1616. URL: `http://papers.nips.cc/paper/3626-a-convergent-on-temporal-difference-algorithm-for-off-policy-learning-with-linear-function-approximation.pdf`.

[12] Yue Wang et al. "Finite Sample Analysis of the GTD Policy Evaluation Algorithms in Markov Setting". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 5504–5513. URL: `http://papers.nips.cc/paper/7134-finite-sample-analysis-of-the-gtd-policy-evaluation-algorithms-in-markov-setting.pdf`.