A Two Time-Scale Update Rule Ensuring Convergence of Episodic Reinforcement Learning Algorithms at the Example of RUDDER

Markus Holzleitner **Jose A. Arjona-Medina** **Marius-Constantin Dinu**
Andreu Vall **Lukas Gruber** **Sepp Hochreiter**
LIT AI Lab
Institute for Machine Learning
Johannes Kepler University Linz, Austria

Abstract

We prove under commonly used assumptions the convergence of episodic, that is, sequence-based, actor-critic-like reinforcement algorithms for which the policy becomes more greedy during learning. The most prominent example of such algorithms is the recently introduced RUDDER method, which speeds up the learning of delayed reward problems by reward redistribution. RUDDER is based on simultaneously learning a policy and a reward redistribution network similar to actor-critic methods. We show the convergence of RUDDER which can be generalized to similar actor-critic-like algorithms. In contrast to previous convergence proofs for actor-critic-like methods, we consider whole episodes as learning examples, undiscounted reward, and a policy that becomes more greedy during learning. We employ recent techniques from two time-scale stochastic approximation theory which are equipped with a controlled Markov process to account for the policy getting more greedy. We expect our framework to be useful to prove convergence of other algorithms based on reward shaping or on attention mechanisms.

1 Introduction

In this work we focus on convergence proofs for actor-critic-like learning methods based on episodic examples, that is, MDPs with finite horizon or absorbing states. In particular we consider rewards that are not discounted. Furthermore we are interested in policies that are "greedy in the limit with infinite exploration" (GLIE) [20]. GLIE policies change toward greediness with respect to the Q-values during learning and allow to prove convergence of tabular SARSA to the optimal policy [20]. GLIE is typically realized by a softmax with exploration coefficient on the Q-values. Our main focus is on proving convergence of learning algorithms for which whole episodes serve as samples. Such methods keep the dependencies between actions and their delayed rewards within episodes. Only if whole episodes are kept and analyzed, long-term dependencies between rewards and the actions which caused them can be detected. The most prominent approach based on learning on whole episodes is RUDDER with overwhelming success for delayed rewards [3, 2]. RUDDER allows for reward redistribution which gives a new redistributed reward to each time point after having seen the reward for the whole episode. Another method that considers the whole episode is Temporal Value Transport (TVT) which uses an attentional memory mechanism to learn a value function that serves as fictitious reward [7].

Convergence of actor-critic algorithms have already been shown using the two time-scale stochastic approximation framework [15, 14, 18, 9]. They mainly rely on techniques summarized e.g. in [4]. However all these approaches are based on state-actions samples or samples of their transitions. We aim at generalizing these proofs to a learning setting where episodes serve as samples. Therefore the idea of stationary distributions on state-action pairs [15, 14] does not apply. We also generalize these methods by allowing the policy becoming more greedy over time. Consequently the two time-scale stochastic approximation framework has to be enriched by a controlled Markov process which describes how the policy becomes more greedy. To the best of our knowledge, this is the first time a two time-scale approximation scheme with an additional control was applied to ensure convergence of a reinforcement learning setting where episodes serve as samples and where the policy is allowed to become more greedy over time. We only prove convergence but not convergence to an optimal policy. Such proofs are in general difficult, since locally stable attractors may not correspond to optimal policies [19, 8, 17]. However, convergence to an locally optimal policy can be proven for linear approximation to Q-values [21, 16].

Our article is structured as follows: the next section very briefly introduces the notation and concepts used for RUDDER [3, 2]. The third section starts with an overview on the results from stochastic approximation theory described in [4] and [9]. Subsequently the setting introduced in the second section is cast in this framework, so that it is also compatible with practical applications. Section 4 finishes with a rigorous proof for the assumptions needed to ensure convergence to a local optimum.

2 Sequence-based Reinforcement learning at the example of RUDDER

2.1 The RUDDER Algorithm

RUDDER is a novel reinforcement learning approach for delayed rewards in finite Markov decision processes (MDPs). We briefly outline its main ideas, for further details consider the original paper [3, 2]. In MDPs the Q-values are equal to the expected immediate reward plus the expected future rewards. Estimating the latter leads to bias problems in temporal difference (TD) learning and to high variance problems in Monte Carlo (MC) learning. These problems are even more severe when one has to deal with delayed rewards. RUDDER aims at making the expected future rewards zero, which simplifies Q-value estimation to computing the mean of the immediate reward. This is realized with the following two new concepts:

- Reward redistribution that leads to return-equivalent decision processes with the same optimal policies and, when optimal, zero expected future rewards.
- Return decomposition via contribution analysis which transforms the reinforcement learning task into a regression task at which deep learning excels.

It turns out that RUDDER is significantly faster than previously used methods in this setting.

We introduce the RUDDER algorithm as a minimization problem of square losses defined below, while sticking to the notation proposed in [3, 2]:

$$\mathrm{L}_h(\boldsymbol{\theta}_n, \boldsymbol{\omega}_n) \;=\; \mathrm{E}_{\breve{\pi}} \left[\frac{1}{2} \sum_{t=0}^{T} \left(R_{t+1}((s,a)_{0:T}; \boldsymbol{\omega}_n) \;-\; \hat{q}(s_t, a_t; \boldsymbol{\theta}_n) \right)^2 \right] \tag{1}$$

$$h(\boldsymbol{\theta}_n, \boldsymbol{\omega}_n) \;=\; \nabla_{\theta_n} \mathrm{L}_h(\boldsymbol{\theta}_n, \boldsymbol{\omega}_n) \tag{2}$$

$$\mathrm{L}_g(\boldsymbol{\theta}_n, \boldsymbol{\omega}_n, z_n) \;=\; \mathrm{E}_{\pi(\boldsymbol{\theta}_n, z_n)} \left[\frac{1}{2} \left(\sum_{t=0}^{T} \tilde{R}_{t+1} \;-\; g((s,a)_{0:T}; \boldsymbol{\omega}_n) \right)^2 \right] \tag{3}$$

$$f(\boldsymbol{\theta}_n, \boldsymbol{\omega}_n, z_n) \;=\; \nabla_{\omega_n} \mathrm{L}_g(\boldsymbol{\theta}_n, \boldsymbol{\omega}_n, z_n) \;, \tag{4}$$

where:

- g denotes the return decomposition function, which is parametrized by $\boldsymbol{\omega}_n \in \mathbb{R}^m$. It predicts the return $\sum_{t=0}^{T} \tilde{R}_{t+1}$ given a complete state-action sequence $(s,a)_{0:T} = \big(s_0, a_0, s_1, a_1, \ldots, s_T, a_T\big)$. In practise g is an LSTM-network.
- $\tilde{R}$ is the original reward provided by the environment.
- $R(\boldsymbol{\omega}_n)$ is the redistributed reward based on the return decomposition of g with parameter vector $\boldsymbol{\omega}_n$. For a state-action sequence $(s,a)_{0:T}$ the realization of its redistributed reward R is computed from $g((s,a)_{0:T}; \boldsymbol{\omega}_n)$ and the realization of return variable $\sum_{t=0}^{T} \tilde{R}_{t+1}$.

- $\hat{q}(s, a; \boldsymbol{\theta}_n)$ is a function parametrized by $\boldsymbol{\theta}_n \in \mathbb{R}^d$ that approximates the Q-value $q(s, a) = r(s, a) = \mathrm{E}\left[R \mid s, a\right]$. We will specify its precise properties later.
- $\breve{\pi}$ is a behavioral policy that doesn't have to depend on the parameters.
- z_n denotes an additional controlled Markov process with values in the compact set $[1, \beta]$ for some $\beta > 1$, whose value will be determined later. We will also postpone the precise construction of this process.
- $\pi(\boldsymbol{\theta}_n, z_n)$ is the policy that is learned by $\hat{q}$ and is updated in every time-step. It will additionally depend on z_n. We will describe the precise construction later.
- $\mathrm{E}_{\pi(\boldsymbol{\theta}_n, z_n)}\left[.\right]$ ($\mathrm{E}_{\breve{\pi}}\left[.\right]$) is an expectation over sequences $(s_0, a_0, s_1, a_1, \ldots, s_t, a_T)$, which are sampled with policy $\pi(\boldsymbol{\theta}_n, z_n)$ ($\breve{\pi}$).

The overall aim is to minimize (1) and (3) by stochastic gradient descent. We continue to describe the formal setup in more detail, and to this end we start by showing that if the policy is sufficiently greedy, the optimal policy can be deduced.

2.2 Sufficient Greediness to Deduce the Optimal Policy

The RUDDER algorithm makes the policy more greedy during learning. We show that we can estimate how greedy the policy must become in order to deduce the optimal policy. We have to guess the minimal difference of the expected return between following the optimal policy and another deterministic policy across all state-action pairs.

The optimal policy will be denoted by π^*, which we assume to be deterministic and unique. It will be approximated by a softmax policy π^+ with bounded parameter β. The quality of approximation is determined by the difference between $v_0^{\pi^+}$ and $v_0^{\pi^*}$, where $v_0^{\pi^+}$ and $v_0^{\pi^*}$ are the according value functions at time $t = 0$. Here we very briefly sketch the main steps towards the desired estimation:

- Due to our assumptions on π^*, each probability $\pi^*(a^i \mid s)$ is either one or zero and the differences $q^{\pi*}_{\max}(s) - q^{\pi*}(s, a^i)$ between Q-values of the best action $q^{\pi*}_{\max}(s)$ and other Q-values $q^{\pi*}(s, a^i)$ ($i \neq \arg\max_i q^{\pi*}(s, a^i)$) have a minimum over s that is larger than zero.
- Next we choose an ϵ as an upper bound on the absolute differences between action-values and values of π^* and π^+: $\left| q^{\pi^*}(s_t, a_t) \ - \ q^{\pi^+}(s_t, a_t) \right| \ < \ \epsilon$ Therefore, for $i_{\max} = \arg\max_i q^{\pi*}(s, a^i)$, we assume that ϵ is small enough to ensure
$$\epsilon \ < \ 1/4 \min_{s, i \neq i_{\max}} q^{\pi*}_{\max}(s) - q^{\pi*}(s, a^i). \tag{5}$$
- Furthermore we assume an approximation $\hat{q}(s, a^i)$ to $q^{\pi^+}(s, a^i)$ which fulfills $\left| \hat{q}(s, a^i) \ - \ q^{\pi^+}(s, a^i) \right| \ < \ \epsilon$.
- The policy π^+ is now defined via the softmax function with approximations $\hat{q}(s, a^i)$:
$$\pi^+(a^i \mid s) \ = \ \frac{\exp(\beta \ \hat{q}(s, a^i))}{\sum_j \exp(\beta \ \hat{q}(s, a^j))} \ = \ \frac{\exp(\beta \ (\hat{q}(s, a^i) \ - \ \hat{q}_{\max}(s)))}{\sum_j \exp(\beta \ (\hat{q}(s, a^j) \ - \ \hat{q}_{\max}(s)))} \ .$$
- One can now conclude after some short calculations:
$$\left| \prod_{t=1}^{T} \pi^*(a_t \mid s_t) \ - \ \prod_{t=1}^{T} \pi^+(a_t \mid s_t) \right| \ < \ T \ (|\mathcal{A}| - 1) \ \exp(- \ \beta \ \delta) \ ,$$
where $|\mathcal{A}|$ is the number of actions.
- Now let K_R be an upper bound on the absolute R_t with $|R_t| < K_R$ and assume that:
$$\beta \ > \max \left(\left(- \ln \frac{\epsilon}{T \ (|\mathcal{A}| - 1) \ |\mathcal{S}|^T \ |\mathcal{A}|^T \ (T + 1) \ K_R} \right) / \delta \ , \ 1 \right) , \tag{6}$$
where $|\mathcal{S}|$ is the number of states and $\delta \ < \ \min_{s, i \neq i_{\max}} \hat{q}_{\max} \ - \ \hat{q}(s, a^i)$.
- Using the mean value theorem we infer after some calculations: $\left| v_0^{\pi^*} \ - \ v_0^{\pi^+} \right| < \epsilon$.

3 The stochastic Approximation Framework

3.1 The General Setting

This section provides a brief summary of the required results from stochastic approximation theory. For further details consider [9] and [4]. An extensive and modern standard reference for the required ODE-theory is [22].

Two-time scale stochastic approximation algorithms are two coupled iterations with different step sizes. For proving convergence of these interwoven iterates it is assumed that one step size is considerably smaller than the other. The slower iterate (the one with smaller step size) is assumed to be slow enough to allow the fast iterate converge while being perturbed by the the slower. The perturbations of the slow should be small enough to ensure convergence of the faster. The iterates map at time step $n \geqslant 0$ the fast variable $\boldsymbol{w}_n \in \mathbb{R}^k$ and the slow variable $\boldsymbol{\theta}_n \in \mathbb{R}^m$ to their new values:

$$\boldsymbol{\theta}_{n+1} \;=\; \boldsymbol{\theta}_n \;+\; a(n)\;\left(h\big(\boldsymbol{\theta}_n, \boldsymbol{w}_n, \boldsymbol{Z}_n^{(1)}\big) \;+\; \boldsymbol{M}_n^{(1)}\right)\;, \tag{7}$$

$$\boldsymbol{w}_{n+1} \;=\; \boldsymbol{w}_n \;+\; b(n)\;\left(f\big(\boldsymbol{\theta}_n, \boldsymbol{w}_n, \boldsymbol{Z}_n^{(2)}\big) \;+\; \boldsymbol{M}_n^{(2)}\right)\;, \tag{8}$$

where:

- $h(.) \in \mathbb{R}^m$ and $f(.) \in \mathbb{R}^k$: mappings for (7) and (8) respectively,
- $a(n)$ and $b(n)$: step sizes for (7) and (8) respectively,
- $\boldsymbol{M}_n^{(1)}$ and $\boldsymbol{M}_n^{(2)}$: additive random Markov processes for (7) and (8) respectively,
- $\boldsymbol{Z}_n^{(1)}$ and $\boldsymbol{Z}_n^{(2)}$: random Markov processes for (7) and (8) respectively.

To ensure convergence of these algorithms, further technical assumptions on the building blocks mentioned before have to be made. The details are mentioned in (A1)–(A7) in [9], where the interested reader can look them up. We will verify these assumptions later for our specific setting.

3.2 More Details and More Precise Formulation

Here we give a more rigorous formulation of the minimization problem introduced in section (2.1), that fits into the framework which we briefly discussed before. Let us start by describing the sampling process, that induces the randomness, in several steps:

- First we fix a baseline probability space: $\Omega = [0,1]$, $P = \lambda$, $\mathcal{A} = \mathcal{B}[0,1]$, with λ denoting the usual Lebesgue measure and $\mathcal{B}[0,1]$ is the Borel σ-algebra on $[0,1]$
- Next let us fix a policy π and introduce the space
$$\tilde{\Omega}_\pi = \{\tau = (s,a)_{0:T} | \tau \text{ is chosen according to } \pi, S_0 = s_0, A_0 = a_0\}\,.$$
The power set $\mathbb{P}(\tilde{\Omega}_\pi)$ serves as the corresponding σ-algebra $\tilde{\mathcal{A}}_\pi$.
- We can endow $\tilde{\mathcal{A}}_\pi$ with a probability measure $\tilde{P}_\pi$, since the probability of choosing a sequence τ with starting point (s_0, a_0) can be computed explicitely. $\tilde{\mathcal{A}}_\pi$ can also be ordered according to the magnitude of the values of its events on P_π. By abuse of notation we denote this ordering again by $\leq$.
- Furthermore we define $S_\pi : \Omega \to \tilde{\Omega}_\pi$ as $S_\pi : x \mapsto \operatorname{argmax}_{\tau \in \tilde{\Omega}_\pi} \left\{ \sum_{\eta \leq \tau} \tilde{P}_\pi(\eta) \leq x \right\}$. This map can easily be seen to be well defined and measurable and it describes how to get one sample from a multinomial distribution with probabilities $\tilde{P}_\pi(\tau)$, where $\tau \in \tilde{\Omega}_\pi$.
- We next describe how to minimize (3) and (1) with stochastic gradient descent: We use an on-line update, where we sample only one state-action sequences, i.e. we introduce functions $\hat{h}$ and $\hat{f}$, where $\hat{h}$ approximates h by using one sample instead of the expectation and similarly for $\hat{f}$. For each time-step n, in $\hat{h}$ the sample is chosen according to the behavioral policy $\breve{\pi}$, whereas for $\hat{f}$ this is done according to $\pi(\boldsymbol{\theta}_n, z_n)$.
- Now set $\boldsymbol{M}_{n+1}^{(1)} = \hat{h}(\boldsymbol{\theta}_n, \boldsymbol{\omega}_n) - h(\boldsymbol{\theta}_n, \boldsymbol{\omega}_n)$ and $\boldsymbol{M}_{n+1}^{(2)} = \hat{f}(\boldsymbol{\theta}_n, \boldsymbol{\omega}_n, z_n) - f(\boldsymbol{\theta}_n, \boldsymbol{\omega}_n, z_n)$.

The rest of the building blocks needed to apply the results from [9] will be explained next.

3.3 Definition of Markov process

In applications the policy π^+ is approximated by policies depending on parameters $z_n \to \beta$ in order to keep up exploration(β usually isn't known at the beginning). We introduce it as a Markov process defined by $z_1 = 1$ and $z_{n+1} = (1 - \frac{1}{\beta})z_n + 1$. The z_n's are nothing more but the partial sums of a geometric series converging to $\beta > 1$ defined in (6). The sequence z_n can be seen as a time-homogeneous Markov process with transition probabilities given by

$$P(\boldsymbol{z}, \boldsymbol{y}) = \delta_{(1-\frac{1}{\beta})\boldsymbol{z}+1}, \tag{9}$$

where δ denotes the Dirac measure. We use the standard notation for discrete time Markov processes, described in detail e.g. in [6]. The approximating policies $\pi(a^i \mid s, \boldsymbol{\theta}, z)$ are given by softmax-functions with parameters z, similarly as in (6).

For proofs in the next chapters we will also need the derivatives of the softmax policies $\pi(z)$, which can easily seen to be bounded.

4 Proofs

4.1 Assumptions on the losses

We assume the following properties of the corresponding loss functions:

- $\hat{q}$, g and R all have compact support and are smooth (say three times continuously differentiable at least) wrt. their parameters.
- For each fixed $\boldsymbol{\theta}$, all critical points of $L_g(\boldsymbol{\theta}, \boldsymbol{\omega}, \beta)$ are isolated. There are only finitely many. For ease of notation we will skip writing down the dependence on β for the moment. The local minima $\{\lambda_i(\boldsymbol{\theta})\}_{i=1}^{k(\boldsymbol{\theta})}\}$ of $L_g(\boldsymbol{\theta}, .)$ can be expressed locally as smooth(say twice continuously differentiable at least) functions with associated compact domains of definitions $\{V_{\lambda_i}\}_{i=1}^{k(\boldsymbol{\theta})}$.
- Locally in V_{λ_i}, $L_h(\boldsymbol{\theta}, \lambda_i(\boldsymbol{\theta}))$, only has one local minimum.

Before proceeding further, let us state some remarks concerning the practical relevance of our assumptions as well as their role in the proof of our result:

- The compact support assumption is compatible with practical applications, since the parameter space of neural networks usually is bounded.
- By the assumption that critical points are isolated, for each starting point $(\boldsymbol{\theta}_0, \boldsymbol{\omega}_0)$ it is possible to find a neighbourhood $G_{\boldsymbol{\theta}_0}(\boldsymbol{\omega}_0)$ that connects $\boldsymbol{\omega}_0$ with a local minimum $\lambda_i(\boldsymbol{\theta}_0)$ of $L_g(., \boldsymbol{\theta}_0)$, and on the other hand such that it avoids saddle points (e.g. if we mainly follow the steepest descent path starting from $\boldsymbol{\omega}_0$ except if it traverses saddles). $\lambda_i(\boldsymbol{\theta}_0)$ is the only local minimum there.
- Moreover, using the implicit function theorem(IFT), we can find a neighbourhood V_0 around $\boldsymbol{\theta}_0$, such that $\lambda_i(\boldsymbol{\theta})$ is twice continuously differentiable and it denotes a local minimum of $L_g(., \boldsymbol{\theta})$(the IFT can be applied to $f(\boldsymbol{\theta}, .) = \nabla_{\boldsymbol{\omega}} L_g(\boldsymbol{\theta}, .) = 0$, since the associated Hessian is positive definite. It can even be shown that it is twice continuously differentiable, using analytic versions of the IFT.)
- In a similar vain for each $\boldsymbol{\theta} \in V_0$ we can now construct neighbourhoods $G_{\boldsymbol{\theta}}(\boldsymbol{\omega}_0)$ around $\boldsymbol{\omega}_0$ with $\lambda_i(\boldsymbol{\theta})$ as unique associated local minimum (we might have to shrink V_0 for this purpose).
- Define $\cup_{\boldsymbol{\theta} \in V_0}(\{\boldsymbol{\theta}\} \times G_{\boldsymbol{\theta}}(\boldsymbol{\omega}_0)) = V_0 \times U_0$. By using a suitable regularization for the networks associated to L_g and L_h, for $(\boldsymbol{\theta}_0, \boldsymbol{\omega}_0)$, we can assume that the algorithm (7)–(8) always stays inside of $V_0 \times U_0$. This, at least heuristically, justifies that for $(\boldsymbol{\theta}_0, \boldsymbol{\omega}_0)$ we localize to $V_0 \times U_0$, as the whole dynamics of the algorithm is determined by this set and we will apply the global results from section (3.1).
- A rigorous justification of this argument would require e.g. a more thorough analysis of stochastic gradient descent, which would of course be a very interesting future research direction.

- It is also a widely accepted (but yet unproven, at least in most practical situations) conjecture, that the probability of ending in a poor local minimum is very small for sufficiently large networks, see e.g. [5, 10, 13, 11, 12], thus we can ensure that (7)–(8) really converges to a useful quantity (a high quality local minimum), if our networks are large enough.

4.2 Proof Details

Now we are in the position to verify the assumptions from section (3.1) needed to ensure convergence of (7)–(8) for the RUDDER-algorithm. As discussed in the previous remarks, we fix a starting point $(\boldsymbol{\theta}_0, \boldsymbol{\omega}_0)$ and focus on the associated neighbourhood $V_0 \times U_0$. This local setting can be replaced by the global one described in (3.1). For ease of notation we omit writing down the dependence on these initial points.

(A1) *Assumptions on the controlled Markov processes:* z_n clearly is a time-homogeneous Markov process with δ_β as its unique invariant measure. So integrating wrt. this invariant measure will in our case just correspond to obtaining the policy π^+.

(A2) *h and f are Lipschitz:* By the mean value theorem it's enough to show that the derivatives wrt. $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ are bounded uniformly wrt. $z \in [1, \beta]$. This, however can be readily deduced from the assumptions from section (4.1) in combination with the boundedness of the softmax-derivative(cf. the explicit expressions (3)–(4)).

(A3) *Martingale difference property and estimates:* From the results in section (3.2) $\boldsymbol{M}_{n+1}^{(1)}$ and $\boldsymbol{M}_{n+1}^{(2)}$ can then easily be seen to be martingale difference sequences wrt. $\mathcal{F}_n$. It remains to show that $\mathrm{E}[|\boldsymbol{M}_{n+1}^{(i)}|^2 | \mathcal{F}_n] \leq B_i$ for $i = 1, 2$. This, however, is also clear, since all the involved expressions are bounded uniformly again by the assumptions from section (4.1).

(A4) *Stepsizes*: We will take the assumptions on the step sizes stated in [9] for granted.

(A5) *Transition kernels:* This is clear by the definition of the transition probabilities, cf. (9).

(A6) *Stability properties of the ODEs:*

- First we introduce $f(\boldsymbol{\theta}, \boldsymbol{\omega}) = f(\boldsymbol{\theta}, \boldsymbol{\omega}, \beta)$ and
$$\dot{\boldsymbol{\omega}}(t) = f(\boldsymbol{\theta}, \boldsymbol{\omega}(t)), \tag{10}$$
where $\boldsymbol{\theta}$ is fixed. (10) can be seen as a gradient system for the function L_g. By standard results on gradient systems (cf. e.g. [1, Section 4] for a nice summary), which guarantee equivalence between strict local minima of the loss function and asymptotically stable points of the associated gradient system, we can use the assumptions and remarks from section (4.1) to ensure that there exists a unique asymptotically stable equilibrium $\lambda(\boldsymbol{\theta})$ of (10).
- The fact that $\lambda(\boldsymbol{\theta})$ is smooth enough can be deduced by the IFT as discussed before.
- For (10) $L_g(\boldsymbol{\theta}, \boldsymbol{\omega}) - L_g(\boldsymbol{\theta}, \lambda(\boldsymbol{\theta}))$ can be taken as associated Lyapunov function $V_{\boldsymbol{\theta}}(\boldsymbol{\omega})$, and thus $V_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ clearly is differentiable wrt. $\boldsymbol{\omega}$ for any $\boldsymbol{\theta}$.
- The slow ODE $\dot{\boldsymbol{\theta}}(t) = h(\boldsymbol{\theta}(t), \lambda(\boldsymbol{\theta}(t))$ also has a unique asymptotically stable fixed point, which again is guaranteed by our assumptions and the standard results on gradient systems.

(A7) *Stability/Boundedess of the iterates:* This is clear by the assumptions in section (4.1).

5 Convergence Theorem

We summarize our findings in the following theorem:

Theorem 1. *Fix a starting point $(\boldsymbol{\theta}_0, \boldsymbol{\omega}_0)$ and determine the associated neighbourhood $U_0 \times V_0$ as in section (4.1). Under the assumptions and remarks stated there the RUDDER algorithm converges to a local minimum $(\boldsymbol{\theta}^*(\boldsymbol{\theta}_0, \boldsymbol{\omega}_0), \lambda(\boldsymbol{\theta}^*(\boldsymbol{\theta}_0, \boldsymbol{\omega}_0)))$ of the associated losses (1) and (3):*
$$(\boldsymbol{\theta}_n, \boldsymbol{w}_n) \ \rightarrow \ (\boldsymbol{\theta}^*(\boldsymbol{\theta}_0, \boldsymbol{\omega}_0), \lambda(\boldsymbol{\theta}^*(\boldsymbol{\theta}_0, \boldsymbol{\omega}_0))) \ \ a.s. \quad as \ n \ \rightarrow \ \infty \ .$$

Conclusions. We prove under commonly used assumptions the convergence of RUDDER, a novel reinforcement learning approach for delayed rewards. We expect our framework to be useful to ensure convergence of other algorithms based on reward shaping or on attention mechanisms.

References

[1] P. A. Absil and K. Kurdyka. On the stable equilibrium points of gradient systems. *Systems & Control Letters*, 55(7):573–577, 2006.

[2] J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter. Rudder: Return decomposition for delayed rewards. *ArXiv*, 2018.

[3] J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter. Rudder: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems 33*, 2019. ArXiv 1806.07857.

[4] V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*, volume 48 of *Texts and Readings in Mathematics*. Springer, 2008.

[5] A. Choromanska et al. The loss surfaces of multilayer networks. *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 192–204, 2015.

[6] M. Hairer. Ergodic properties of markov processes. *Lecture notes*, 2018.

[7] C. Hung, T. Lillicrap, J. Abramson, Y. Wu, M. Mirza, F. Carnevale, A. Ahuja, and G. Wayne. Optimizing agent behavior over long time scales by transporting value. *ArXiv*, 2018.

[8] C. Jin, P. Netrapalli, and M. I. Jordan. Minmax optimization: Stable limit points of gradient descent ascent are locally optimal. *ArXiv*, 2019.

[9] P. Karmakar and S. Bhatnagar. Two time-scale stochastic approximation with controlled Markov noise and off-policy temporal-difference learning. *Mathematics of Operations Research*, 2017.

[10] K. Kawaguchi. Deep learning without poor local minima. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 586–594, 2016.

[11] K. Kawaguchi and Y. Bengio. Depth with nonlinearity creates no bad local minima in ResNets. *Neural Networks*, 118:167–174, 2019.

[12] K. Kawaguchi, J. Huang, and L. P. Kaelbling. Effect of depth and width on local minima in deep learning. *Neural Computation*, 31(6):1462–1498, 2019.

[13] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

[14] V. R. Konda and V. S. Borkar. Actor-critic-type learning algorithms for Markov decision processes. *SIAM J. Control Optim.*, 38(1):94–123, 1999.

[15] V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, pages 1008–1014, 2000.

[16] V. R. Konda and J. N. Tsitsiklis. On actor-critic algorithms. *SIAM J. CONTROL OPTIM.*, 42(4):1143–1166, 2003.

[17] T. Lin, C. Jin, and M. I. Jordan. On gradient descent ascent for nonconvex-concave minimax problems. *ArXiv*, 2019.

[18] H. R. Maei, C. Szepesvári, S. Bhatnagar, D. Precup, D. Silver, and R. S. Sutton. Convergent temporal-difference learning with arbitrary smooth function approximation. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1204–1212. Curran Associates, Inc., 2009.

[19] E. V. Mazumdar, M. I. Jordan, and S. S. Sastry. On finding local Nash equilibria (and only local Nash equilibria) in zero-sum games. *ArXiv*, 2019.

[20] S. Singh, T. Jaakkola, M. Littman, and C. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38:287–308, 2000.

[21] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, pages 1057–1063, 2000.

[22] G. Teschl. *Ordinary differential equations and dynamical systems*, volume 140. American Mathematical Soc., 2012.