Temporal Regularization for Markov Decision Process

Pierre Thodoroff

Computer Science
McGill University, Montreal

January 29, 2019

Acknowledgements

TODO

Abstract

Several applications of Reinforcement Learning suffer from instability due to high variance. This is especially prevalent in high dimensional domains. Regularization is a commonly used technique in machine learning to reduce variance, at the cost of introducing some bias. Most existing regularization techniques focus on spatial (perceptual) regularization. Yet in reinforcement learning, due to the nature of the Bellman equation, there is an opportunity to also exploit temporal regularization based on smoothness in value estimates over trajectories. This paper explores a class of methods for temporal regularization. We formally characterize the bias induced by this technique using Markov chain concepts. We illustrate the various characteristics of temporal regularization via a sequence of simple discrete and continuous MDPs, and show that the technique provides improvement even in high-dimensional Atari games.

Résumé

Several applications of Reinforcement Learning suffer from instability due to high variance. This is especially prevalent in high dimensional domains. Regularization is a commonly used technique in machine learning to reduce variance, at the cost of introducing some bias. Most existing regularization techniques focus on spatial (perceptual) regularization. Yet in reinforcement learning, due to the nature of the Bellman equation, there is an opportunity to also exploit temporal regularization based on smoothness in value estimates over trajectories. This paper explores a class of methods for temporal regularization. We formally characterize the bias induced by this technique using Markov chain concepts. We illustrate the various characteristics of temporal regularization via a sequence of simple discrete and continuous MDPs, and show that the technique provides improvement even in high-dimensional Atari games.

Contents

Contents **iv**

1 Introduction **1**

1.1 Intro part 2 2

2 Markov Chains **3**

2.1 Discrete-time Markov chains 3

2.2 Stationary distribution 5

2.3 Detailed Balance 5

2.4 Mixing Time 6

2.5 Reversal Markov Chains 6

3 Reinforcement Learning **8**

3.1 Markov Decision Process 8

3.2 Policy Evaluation 9

3.2.1 Bellman Operator 10

3.2.2 Temporal Difference 11

3.2.3 Lambda Return 13

3.2.4 Linear Function Approximation 14

3.3 Control 14

3.3.1 Bellman Optimilatiy Equations 14
3.3.2 Policy Iteration 15
3.3.3 Q function 16
3.3.4 Policy Gradient 16
3.3.5 Actor-Critic 17
3.4 Deep Reinforcement Learning 17
3.4.1 A3C 18
3.4.2 PPO 18

4 Regularization in Markov Decision Process 20
4.1 Regularization 20
4.1.1 Spatial Regularization 20
4.1.2 Policy Regularization 22
4.1.3 Temporal Regularization 22
4.1.4 Deliberation cost 23

5 Temporal Regularization 24
5.1 Value Based Temporal Regularization 24
5.1.1 Definition 24
5.1.2 Bias 26
5.1.3 Variance 27
5.1.4 Average reward case: 27
5.1.5 Temporal Regularization as a time series prediction problem: . 27
5.1.6 Control 28
5.1.7 Temporal difference with function approximation 28
5.1.8 Connection with multi-step methods 30
5.2 Experiments 30
5.2.1 Mixing time 30
5.2.2 Bias 30

5.2.3 Variance number 2 32
5.2.4 Variance 32
5.2.5 Propagation of the information 33
5.2.6 Noisy state representation 34
5.2.7 Deep reinforcement learning 36
5.3 Policy Gradient Temporal Regularization 37

6 Recurrent Learning 40
6.1 Introduction 40
6.2 Technical Background 43
6.3 Recurrent Learning in Reinforcement Learning 44
6.3.1 Asymptotic convergence 46
6.3.2 State dependent β 50
6.3.3 Adjusting for the reward 53
6.4 Related work 53
6.5 Experiments 54
6.5.1 Bias Variance trade-off 55
6.5.2 Partially observable setting 56
6.5.3 Deep reinforcement learning 57
6.6 Discussions 61
6.6.1 β as an interest function: 61
6.6.2 Partial observability: 61
6.6.3 Relationship between λ and β: 62
6.6.4 Asymptotic convergence: 62
6.6.5 Bias of the eligibility trace: 62
6.6.6 Recurrent learning for action: 63

7 Conclusion 64

Bibliography **65**

1

Introduction

There has been much progress in Reinforcement Learning (RL) techniques, with some impressive success with games Silver et al. 2016, and several interesting applications on the horizon Koedinger et al. 2018; Shortreed et al. 2011; Prasad et al. 2017; Dhingra et al. 2017. However RL methods are too often hampered by high variance, whether due to randomness in data collection, effects of initial conditions, complexity of learner function class, hyper-parameter configuration, or sparsity of the reward signal Henderson et al. 2017. Regularization is a commonly used technique in machine learning to reduce variance, at the cost of introducing some (smaller) bias. Regularization typically takes the form of smoothing over the observation space to reduce the complexity of the learner's hypothesis class.

In the RL setting, we have an interesting opportunity to consider an alternative form of regularization, namely temporal regularization. Effectively, temporal regularization considers smoothing over the trajectory, whereby the estimate of the value function at one state is assumed to be related to the value function at the state(s) that typically occur before it in trajectories. This structure arises naturally out of the fact that the value at each state is estimated using the Bellman equation. The standard Bellman equation clearly defines the dependency between value estimates. In temporal regularization we amplify this dependency by making each state depend more strongly on estimates of *previous* states as opposed to multi-step that considers

future states.

This paper proposes a class of temporally regularized value function estimates. We discuss properties of these estimates, based on notions from Markov chains, under the policy evaluation setting and extend the notion to the control case. Our experiments show that temporal regularization effectively reduces variance and estimation error in discrete and continuous MDPs. The experiments also highlight that regularizing in the time domain rather than in the spatial domain allows more robustness to cases where state features are mis-specified or noisy, as is the case in the Atari domains.

1.1 Intro part 2

Graph smooth estimate like time series

- Fitting a model
- averaging of previous values
- Entropy term

2

Markov Chains

2.1 Discrete-time Markov chains

We begin by introducing discrete Markov chain concepts that will be used to study the properties of temporally regularized MDPs. In this thesis, we focus on discrete Markov chains, however the concepts can be extended to the continuous case.

Discrete Markov chain are stochastic models representing sequences of random variable satisfying the Markov property. Formally, we define a discrete-time Markov chain (Norris 1998; Levin and Peres 2008; Brémaud 2013) with finite state space $\mathcal{S}$ by a sequence of $|\mathcal{S}|-$valued random variable $X_0, X_1, ...$ and a transition function $\mathcal{P} : \mathcal{S} \times \mathcal{S} \mapsto [0, 1]$. The sequence of random variable needs to satisfy the Markov property:

Definition 1. *A stochastic process satisfy Markov property if:*

$$\mathcal{P}(X_{n+1} = j | X_n = i, X_{n-1} = k, ...) = \mathcal{P}(X_{n+1} = j | X_n = i) \tag{2.1}$$

Intuitively, this means that the probability of moving to the next states is based solely on its present state and not its history. One of the most studied Markov chains considers the property of randomly walking on a chain as described in figure (2.1).

Discrete time Markov chains can also be represented in matrix form. The transition function can be represented as a $|\mathcal{S}| \times |\mathcal{S}|$ matrix P such that $P_{ij} = \mathcal{P}(X_{n+1} = j | X_n =$

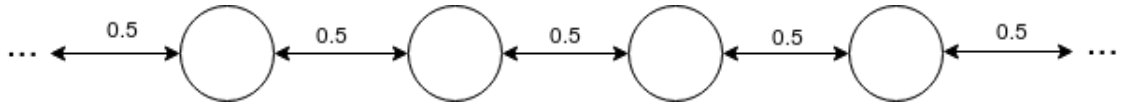


Figure 2.1: Random walk

i). Studying Markov chains from an algebraic perspective can sometimes simplify the analysis.

We now define some useful fundamental properties of Markov chains.

If every state is accessible from every other, the chain(or its transition matrix) is said to be irreducible.

Definition 2. *A chain is said to be irreducible if* $\forall i, j \in \mathcal{S}$ *there exists* $n \in \mathbb{N}$ *such that:*

$$P(X_n = j | X_0 = i) > 0 \tag{2.2}$$

Definition 3. *The period of a state is the greatest common divisor of the set* $n \in \mathbb{N} : P(X_n = i | X_0 = i) > 0$. *The chain is defined as aperiodic if every state has period 1.*

Definition 4. *Let* T_i *define the hitting time of state i such that:*

$$T_i = inf(n \geq 1 : X_n = i | X_0 = i) \tag{2.3}$$

We define a transient state if T_i *is not finite. A state is called recurrent if it is not transient.*

Definition 5. *A chain is defined as ergodic if it is positive recurrent and aperiodic.*

Throughout this thesis, we make the following mild assumption on the Markov chain:

Assumption 1. *P is ergodic*

Most of the Reinforcement learning theory relies on this assumption. However, some works considers the case when the chain is not ergodic (Leike 2016).

2.2 Stationary distribution

It is often interesting to study the properties of Markov chains in the limit. Over the long run, we define the stationary distribution μ_i as the *proportion of time* spend in each state $i \in \mathcal{S}$.

Definition 6. *Assuming that P is ergodic, P has a unique stationary distribution μ that satisfies:*

$$\begin{aligned} \mu &= \mu P \\ \sum_i \mu_i &= 1 \end{aligned} \tag{2.4}$$

There exists many different metrics used to define distance's from stationary distribution (Levin and Peres 2008). One common metric in discrete Markov chains can be defined as follows:

$$d_t(P) = \left\| P^t \mathbb{1} - \mu \right\|_\infty \tag{2.5}$$

where $\mathbb{1}$ is a vector of one's.

2.3 Detailed Balance

The concept of detailed balance originated from physics. It is used to study the behavior of systems in the limit at *equilibrium*.

Definition 7 (Detailed balance Kemeny and Snell 1976)**.** *Let P be an irreducible Markov chain with invariant stationary distribution μ. μ_i defines the ith element of μ. A chain is said to satisfy detailed balance if and only if*

$$\mu_i P_{ij} = \mu_j P_{ji} \qquad \forall i, j \in \mathcal{S}. \tag{2.6}$$

Intuitively, this means that if we start the chain in a stationary distribution, the amount of probability that flows from i to j is equal to the one from j to i. In other

words, the system must be at equilibrium. An intuitive example of a physical system not satisfying detailed balance is a snow flake in a coffee.

Remark. *If a chain satisfies detailed balance, it is called reversible.*

2.4 Mixing Time

In Markov chains theory, one of the main challenge is to study the mixing time of the chain (Levin and Peres 2008). The mixing time corresponds to the time needed for the chain to be *close* to its stationary distribution μ. More formally it can be defined as:

$$t_{mix}(\epsilon) = \min\{t : d_t(P) < \epsilon\} \tag{2.7}$$

where $d_t(P)$ can be defined as in (2.5).

When the chain is reversible, it is possible to estimate and bound the mixing time relatively efficiently (Diaconis and Stroock 1991)

Indeed, many chains do not satisfy this detailed balance property. In this one case it is possible to use a different, but related, chain called the reversal Markov chain to infer mixing time bounds (Fill et al. 1991; K.-M. Chung et al. 2012).

2.5 Reversal Markov Chains

The reversal Markov chain $\widetilde{P}$ can be interpreted as the Markov chain P with time running backwards. It is be a key concept used to define convergence and bias induced by temporal regularization later in this thesis.

Definition 8 (Reversal Markov chain Kemeny and Snell 1976)**.** *Let $\widetilde{P}$ the reversal Markov chain of P be defined as:*

$$\widetilde{P_{ij}} = \frac{\mu_j P_{ji}}{\mu_i} \qquad \forall i, j \in \mathcal{S}. \tag{2.8}$$

As an example, assuming a Markov chain P has a uniform stationary distribution, if a transition is highly irreversible like falling off a cliff ($P_{ij}\#P_{ji}$) the difference between the forward and the backward chain in that state will be high.
We now introduce some properties of reversal Markov chains that will be used later in the thesis.

Remark. *If P is irreducible with invariant distribution μ, then $\widetilde{P}$ is also irreducible with invariant distribution μ.*

Remark. *If P is reversible than $P = \widetilde{P}$*

Furthermore, both P and $\widetilde{P}$ have the same stationary distribution and so does any convex combination of them.

Lemma 1. *P and $(1-\beta)P+\beta\widetilde{P}$ have the same stationary distribution μ* $\quad \forall\beta \in [0,1]$.

Proof.

$$\begin{aligned} \mu((1-\beta)P + \beta\widetilde{P}) &= (1-\beta)\mu P + \beta\mu\widetilde{P} \\ &= (1-\beta)\mu + \beta\mu \\ &= \mu. \end{aligned} \tag{2.9}$$

□

3

Reinforcement Learning

In science, mathematical frameworks are used to study the behavior of objects. In physics, for example, the newton's laws laid the foundation of classical mechanics used to describe the motion of macroscopic objects. In this thesis, we are interested in the problem of sequential decision making. The most popular mathematical framework used to study sequential decision making is called Markov Decision Process(MDP). The underlying core assumption is the Markovian assumption on the state space. MDP assumes the state space is fully observable and the future is independant of the past conditioned on the current state. More formally, this can be described as:

$$p(s_{t+1}|s_t, s_{t-1}, ..., s_0) = p(s_{t+1}|s_t) \tag{3.1}$$

Sequential decision making differs from supervised learning in several ways. Supervised learning is a set of models designed to predict an output based on IID data. However, in reinforcement learning our prediction/decision often impact the distribution of the data. For example, choosing to do turn right at an intersection will siginificantly change the distribution of future states.

3.1 Markov Decision Process

We now formally introduce the concept used in Markov Decision Process.

Definition 9 (Markov Decision Process (Puterman 1994)). *A Markov Decision Process (MDP) is defined as a tuple $(\mathcal{S}, \mathcal{P}, r)$ where:*

- $\mathcal{S}$ *is a discrete set of states*
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ *is a transition function*
- $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ *is a reward function*

On each round t, the learner observes current state $s_t \in \mathcal{S}$ and selects action $a_t \in \mathcal{A}$, after which it receives reward $r_t = r(s_t, a_t)$ and moves to new state $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$. We define a stationary policy π as a probability distribution over actions conditioned on states $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, such that $a_t \sim \pi(\cdot|s_t)$.

3.2 Policy Evaluation

For policy evaluation, given a policy π, the goal is to find the associated value function of that policy v^π. When performing policy evaluation in the discounted case, the goal is to estimate the discounted expected return of policy π at a state $s \in \mathcal{S}$, $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s]$, with discount factor $\gamma \in [0, 1)$. This can be written in matrix form as:

$$v^\pi = \sum_{i=0}^{\infty} \gamma^i (P^\pi)^i r \tag{3.2}$$

where P^π denotes the $|\mathcal{S}| \times |\mathcal{S}|$ transition matrix under policy π, v^π is the state values column-vector, and r is the reward column-vector. The actions do not appear in the equations explicitly as the policy has been coupled inside P^π.The matrix P^π also defines a Markov chain. In practice, we often do not have access to the transitions and rewards directly. Instead, we sample tuples (s, s', r) from the environment and use those to estimate in expectation the discounted expected cumulative return for a state. The most straightforwad way to estimate v would be to collect tuples (s, s', r) and average the discounted reward. However, this often suffers from high variance

(Kearns and S. P. Singh 2000). By unrolling the discounted sum of reward it is possible to obtain a recursive form based on v:

$$\begin{aligned} v^\pi(s_0) &= \mathop{\mathbb{E}}_{\pi}[r(s_0, \pi(s_0)) + \sum_{t=1}^{\infty} \gamma^t r_{t+1} | s_0 = s] \\ &= \mathop{\mathbb{E}}_{\pi}[r(s_0, \pi(s_0)) + \gamma \mathop{\mathbb{E}}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_1] | s_0 = s] \\ &= \mathop{\mathbb{E}}_{\pi}[r(s_0, \pi(s_0)) + \gamma v^\pi(s_1) | s_0 = s] \end{aligned} \tag{3.3}$$

This can also be rewritten as a linear system of equation and solved using standard algebra methods:

$$\begin{aligned} v^\pi &= r + \gamma P^\pi v^\pi \\ &\equiv (I - P^\pi) v^\pi = r \end{aligned} \tag{3.4}$$

However, inverting a matrix can be costly, unstable and as mentioned earlier in practice we often do not have access to P^π, r directly. This suggests that using iterative method may be more suitable for reinforcement learning.

3.2.1 Bellman Operator

We consider the operator-theoretic point of view by defining the following operator

Definition 10. *The Bellman operator $\mathcal{T}^\pi$ has a unique fixed point v^π where:*

$$\mathcal{T}^\pi v = r + \gamma P^\pi v \tag{3.5}$$

In order to show this we use Banach Fixed point theorem stating that:

Theorem 1 (Banach Fixed Point Theorem (Banach 1922))**.** *Let U be a Banach space: if $\mathcal{T}U \to U$ is a contraction mapping then:*

- *There exists a unique fixed point v^* such that $\mathcal{T}v^* = v^*$*
- *$lim_{t\to\infty} \mathcal{T}^t v = v^*$*

As we saw in the previous section in equation 3.4 v^π is a fixed point. We can prove its unicity and the convergence of the operator by proving that $\mathcal{T}^\pi$ is a contraction. In this thesis, unless stated otherwise, the norm considered is the infinity norm.

Lemma 2. *$\mathcal{T}^\pi$ is a contraction with a contraction factor of γ*

Proof.

$$\begin{aligned} \|\mathcal{T}^\pi u - \mathcal{T}^\pi v\| &= \|r + \gamma P^\pi u - (r + \gamma P^\pi v\| \\ &= \|\gamma P^\pi (u - v)\| \\ &= \gamma \|u - v\| \end{aligned} \tag{3.6}$$

□

Algorithm 1 shows an example of a stochastic version of equation 3.5.

Algorithm 1 Policy evaluation

1: Input: π, γ
2: **for all** steps **do**
3: Choose $a \sim \pi(\mathcal{S})$
4: Take action a, observe $r(s), s'$
5: $V(s) = r(s) + \gamma V(s')$
6: **end for**

It is also possible to study the stochastic version of this operator using techniques from dynamical system and ordinary differential equations (Borkar and Meyn 2000).

3.2.2 Temporal Difference

At each step of the bellman operator the previous estimate v_t^π at time step t is forgotten.

$$v_{t+1} = r + \gamma P v_t \tag{3.7}$$

Temporal difference attempts to exploit previous estimates by averaging them such that $v_{t+1} = \alpha \sum_{i=0}^{t+1} (1-\alpha)^{t-i} v_i$

$$\begin{aligned} v_{t+1} &= \alpha(r + \gamma P v_t) + (1-\alpha) v_t \\ &= \alpha(r + \gamma P v_t) + (1-\alpha)[\alpha(r + \gamma P v_{t-1}) + (1-\alpha) v_{t-1}] \end{aligned} \tag{3.8}$$

This gives rise to the temporal difference algorithm:

$$\begin{aligned} v_{t+1} &= \alpha(r + \gamma P v_t) + (1-\alpha) v_t \\ &= \alpha(r + \gamma P v_t - v_t) + v_t \end{aligned} \tag{3.9}$$

Definition 11 (Temporal Difference (R. S. Sutton 1988)). *The temporal difference operator $\mathcal{T}_\alpha$ parametrized by α can be written as:*

$$\mathcal{T}_\alpha v = (1-\alpha) v + \alpha(r + \gamma P v) \tag{3.10}$$

A stochastic version of temporal difference can be found in algorithm 2.

Algorithm 2 Temporal Difference

1: Input: π, α, γ
2: **for all** steps **do**
3: Choose $a \sim \pi(\mathcal{S})$
4: Take action a, observe $r(s), s'$
5: $V(s) = V(s) + \alpha(r(s) + \gamma V(s') - V(s))$
6: **end for**

Bootstrapping on previous estimates may introduce bias depending on how well v is estimated. Several papers attempt to characterize this bias in various ways (Kearns and S. P. Singh 2000; R. S. Sutton and S. P. Singh n.d.). Methods bootstrapping on previous estimates are also called semi-iterative methods (Varga 2009) in the field of iterative methods. It would be interesting to examine algorithms like chebyshev semi-iterative method (Golub and Varga 1961) that attempts to find optimal α's using chebyshev polynomials. There might exists interesting connections with meta-learning algorithms.

3.2.3 Lambda Return

In the previous sections we defined algorithms that bootstrap on the next value, to reduce variance, instead of looking at the rewards. Lambda return (R. S. Sutton 1984) generalizes this intuition by bootstrapping on all future values, not just the direct next one. This is done by first unrolling the bellman updates, yielding N-step return of the form:

Definition 12 (N-step return)**.**

$$\mathcal{T}_n = \sum_{i=0}^{n} \gamma^i r + \gamma^{n+1} P v \tag{3.11}$$

Instead of choosing a specific N, lambda-return exponentially averages through all the N-step return.

Definition 13 (Lambda-return)**.**

$$\mathcal{T}^{(\lambda)} = (1-\lambda) \sum_{i=0}^{\infty} \lambda^i \mathcal{T}_i \tag{3.12}$$

Varying λ yields Monte Carlo on one side ($\lambda \to 1$) and TD(0) on the other ($\lambda \to 0$). Lambda-return can be implemented efficiently in an online fashion using eligibility traces (R. S. Sutton 1984; S. P. Singh and R. S. Sutton 1996; Precup 2000).

Algorithm 3 Temporal Difference with eligibility traces

1: Input: $\pi, \alpha, \gamma, \lambda$
2: **for all** steps **do**
3: Choose $a \sim \pi(\mathcal{S})$
4: Take action a, observe $r(s), s'$
5: $e(t) = \mathbb{1}_s + \gamma\lambda e(t-1)$
6: $V = V + \alpha(r(s) + \gamma V(s') - V(s))e(t)$
7: **end for**

However, this algorithm is biased if the trace is used online. This is due to the fact that when an update is done, the trace becomes biased as the distribution with respect to the new parameters would have been different. True online TD (Seijen and

R. Sutton 2014) solves this issue by accounting for those changes. This problem is similar to the one encountered by real time recurrent learning (Williams and Zipser 1995). It would be interesting to study if True Online TD can be extended to the non-linear settings.

3.2.4 Linear Function Approximation

The methods developed previously scale linearly with the number of states. This becomes quickly intractable for large discrete state space and continuous settings. One way to remedy this is to use function approximation. It is possible to develop similar operators using function approximator $v_\theta^\pi(s)$ which can be written in the linear case as :

$$v_\theta^\pi(s) = \phi\theta \tag{3.13}$$

where ϕ is a vector representing the features of state s. In practice, θ is learned by minimizing the following objective:

$$\mathop{\mathbb{E}}_{\pi}\left(v^\pi(s) - v_\theta^\pi(s)\right)^2 \tag{3.14}$$

where $v^\pi(s)$ is the *target*. The target can be any estimate proposed earlier(Monte-carlo or Boostrap).

3.3 Control

In the previous section we discussed estimating the value of a policy. However, in many cases, the actual goal is to use this estimate to *improve* on the policy. This is called control.

3.3.1 Bellman Optimilatiy Equations

In the control case, the goal is to find the optimal policy π^* that maximizes the discounted expected return. As in the previous section we define the optimal value

function V^* as the fixed point of the non-linear optimal Bellman operator:

$$\mathcal{T}^* v^* = \max_{a \in \mathcal{A}} [r(a) + \gamma P(a) v^*]. \tag{3.15}$$

3.3.2 Policy Iteration

Using the operator defined in eq 3.15, we now define our first control algorithm. The main framework used for control in Reinforcement Learning is called Generalized Policy Improvements(GPI) (R. S. Sutton and Barto 1998). GPI alternates between policy evaluation and policy improvements.

Algorithm 4 General Policy Improvements

```
Input: π, α, γ
Policy Evaluation:
    for all steps do
        Choose a ∼ π(S)
        Take action a, observe r(s), s'
        V(s) = r(s) + γV(s')
    end for
Policy Improvement:
    for all s ∈ S do
        π(s) ← argmax_a Σ_{s'} p(s', r|s, a) + [r + γV(s')]
    end for
If policy stable stop, else go to Policy Evaluation
```

However, this algorithm requires to have a model of the environment to select the next optimal action. To remedy the concept of Q-function was introduced (Watkins and Dayan 1992).

3.3.3 Q function

Q-function are state action pair representing the expected discounted return from s_0 if action a_0 is to be taken.

$$\begin{aligned} Q(s,a) &= \mathop{\mathbb{E}}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s = s_0, a = a_0] \\ &= \mathop{\mathbb{E}}_{s'}[r_t + \gamma v(s')] \\ &= \mathop{\mathbb{E}}_{s'}[r_t + \gamma \max_{a'} Q(s', a')] \end{aligned} \tag{3.16}$$

Selecting the optimal action can easily be done by taking the max Q-values over all actions. This algorithm is called Sarsa(State–action–reward–state–action).

Algorithm 5 SARSA

1: Input: π, α, γ
2: **for all** steps **do**
3: Choose $a \sim \pi(\mathcal{S})$
4: Take action a, observe $r(s), s'$
5: $Q(s,a) = Q(s,a) + \alpha(r(s) + \gamma Q(s', a') - Q(s,a))$
6: **end for**

Estimating the optimal action requires calculating the value of each action and taking the argmax. This can be problematic in continous action space.

3.3.4 Policy Gradient

If the action space is continuous or large, using look-up tables for Q-values can become quickly intractable. One way to circumvent this problem is to use function approximation on the policy. We define a policy π_θ parametrized by theta. The goal is to find a set of parameters θ such as to maximize: $J(\theta) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1}]$.

Definition 14 (Policy Gradient (R. S. Sutton, McAllester, et al. 2000))**.** *The gradient* $\nabla J(\theta)$ *can be expressed as:*

$$\nabla J(\theta) = \int_s d^\pi(s) \int_a \nabla_\theta \pi(a,s) Q^\pi(s,a) \tag{3.17}$$

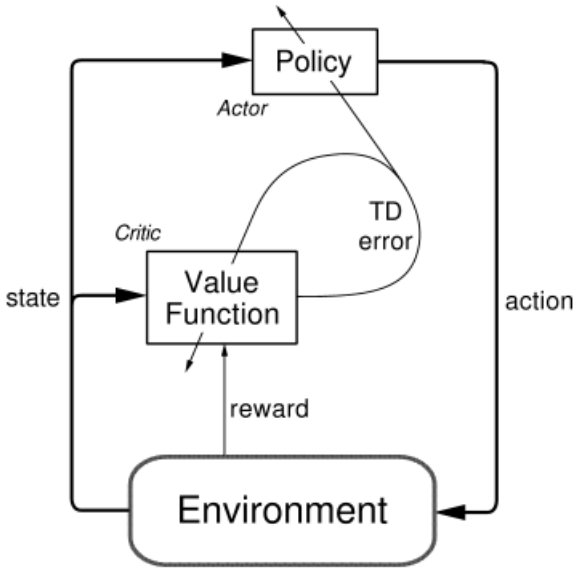


Figure 3.1: Actor-critic

In practice, those integrals can be estimated using samples. Furthermore Q is often unknown, but can be estimated using rollouts G_t to approximate $V(s)$. This algorithm is called REINFORCE (Williams and Zipser 1995).

3.3.5 Actor-Critic

Policy gradient methods can be enhanced by bootstrapping on a learned value function to approximate G_t instead of using Monte Carlo rollouts. This is the central idea behind actor-critic (Konda and Tsitsiklis 2000) method which can be summarized in figure 3.1. Actor critic can be thought of as combining policy gradient and value based method. Many of the recent algorithms developed are based on this framework (Mnih, Badia, et al. 2016; Schulman, Wolski, et al. 2017; Wu et al. 2017).

3.4 Deep Reinforcement Learning

Reinforcement learning algorithms combined with deep neural network as function approximation yields promising results on high dimensional problem (Mnih, Kavukcuoglu,

Silver, Graves, et al. 2013; Mnih, Badia, et al. 2016; Schulman, Wolski, et al. 2017) such as Atari (Bellemare et al. 2013) and Mujoco (Todorov, Erez, and Tassa 2012). However, they appear to be very unstable and requires a number of *tricks* to behave properly. We will describe 2 different algorithm Asynchronous Actor Critic (A3C Mnih, Badia, et al. 2016) and Proximal Policy Optimzation (PPO Schulman, Wolski, et al. 2017) and explain the tools used to make them stable. Both methods are based on actor critic formulation.

3.4.1 A3C

One severe problem when using deep neural network as function approximation arise from the fact that the data is not IID. A3C (Mnih, Badia, et al. 2016) is an actor-critic method developed to overcome this issue when used with deep neural networks. This is effectively done by running several agents in parallel and using a batch of data to update the parameters. This effectively decorrelates the samples seen by the agents. A diagram explaining the algorithm can be found in figure 3.2.

3.4.2 PPO

Another important issue in Reinforcement Learning is the one of non-stationarity of the data. Proximal methods (Schulman, Wolski, et al. 2017; Schulman, Levine, et al. 2015) alleviate this by limiting the update of the actor at each time step:

$$\beta \mathrm{KL}(\pi_{\mathrm{old}}, \pi_{\mathrm{new}}) \tag{3.18}$$

where β control the magnitude of the regularization. This regularization has been found to be effective when combined with deep neural network as function approximator.

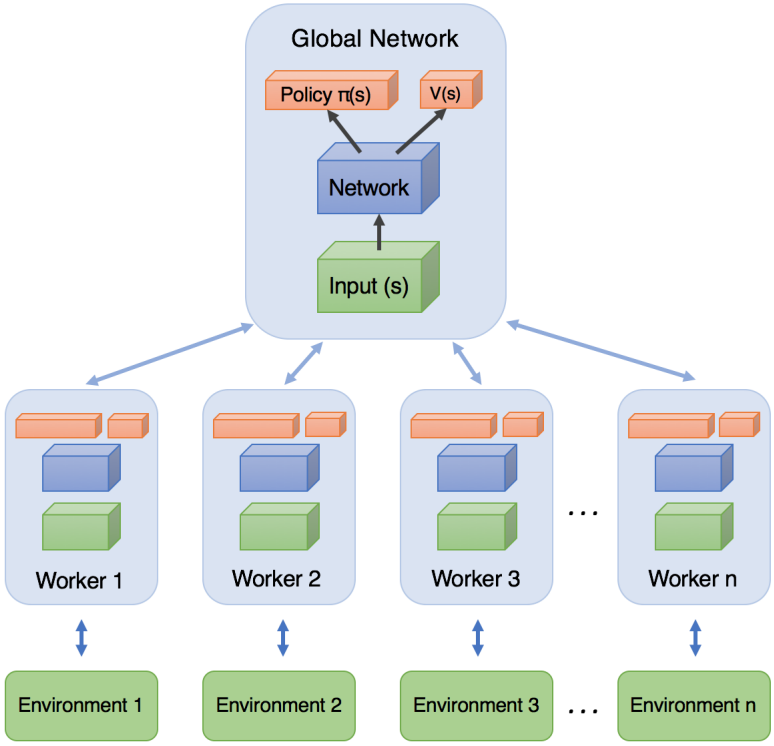


Figure 3.2: A3C diagram from LOOK AT COPYRIGHT

4

Regularization in Markov Decision Process

Regularization in Machine Learning is a cornerstone. Most models can be seen as bias/variance trade-off.

4.1 Regularization

Regularization in RL has been considered in several different perspectives. In this section we discuss the most popular one. Other works focus on regularizing the changes in policy directly. Those approaches are often based on entropy methods (Neu, Jonsson, and Gómez 2017; Schulman, Wolski, et al. 2017; Bartlett and Tewari 2009). Explicit regularization in the temporal space has received much less attention. Temporal regularization in some sense may be seen as a "backward" multi-step method (R. S. Sutton and Barto 1998).

4.1.1 Spatial Regularization

One line of investigation focuses on regularizing the features learned on the state space (Farahmand et al. 2009; Petrik et al. 2010; Pazis and Parr 2011; Farahmand 2011; B. Liu, Mahadevan, and J. Liu 2012; Harrigan 2016). These approaches assume that nearby states in the state space have similar value.

4.1.1.1 Regularization of the function approximation

Farahmand 2011 This thesis studies the reinforcement learning and planning problems that are modeled by a discounted Markov Decision Process (MDP) with a large state space and finite action space. We follow the value-based approach in which a function approximator is used to estimate the optimal value function. The choice of function approximator, however, is nontrivial, as it depends on both the number of data samples and the MDP itself. The goal of this work is to introduce flexible and statistically-efficient algorithms that find close to optimal policies for these problems without much prior information about them. The recurring theme of this thesis is the application of the regularization technique to design value function estimators that choose their estimates from rich function spaces. We introduce regularization-based Approximate Value/Policy Iteration algorithms, analyze their statistical properties, and provide upper bounds on the performance loss of the resulted policy compared to the optimal one. The error bounds show the dependence of the performance loss on the number of samples, the capacity of the function space to which the estimated value function belongs, and some intrinsic properties of the MDP itself. Remarkably, the dependence on the number of samples in the task of policy evaluation is minimax optimal. We also address the problem of automatic parameter-tuning of reinforcement learning/planning algorithms and introduce a complexity regularization-based model selection algorithm. We prove that the algorithm enjoys an oracle-like property and it may be used to achieve adaptivity: the performance is almost as good as the performance of the unknown best parameters. Our two other contributions are used to analyze the aforementioned algorithms. First, we analyze the rate of convergence of the estimation error in regularized least-squares regression when the data is exponentially beta-mixing. We prove that up to a logarithmic factor, the convergence rate is the same as the optimal minimax rate available for the i.i.d. case. Second, we attend to the question of how the errors at each iteration of the approximate pol-

icy/value iteration influence the quality of the resulting policy. We provide results that highlight some new aspects of these algorithms.

4.1.2 Policy Regularization

There exists several ways to regularize policy gradient method. The most comonly used in recent research is called entropy regularization Neu, Jonsson, and Gómez 2017; Schulman, Wolski, et al. 2017; Bartlett and Tewari 2009.

4.1.2.1 Entropy Regularization

In policy gradient method, one successful approach consists of regularizing the entropy of your policy distribution Neu, Jonsson, and Gómez 2017. Policy gradient methods will tend to converge to distributions with low entropy. By adding an entropy bonus, it encourages the policy to explore and often yields much better performance.

$$\begin{aligned} \pi &= \mathop{\mathbb{E}}_{\pi} r + R(\pi) \\ R(\pi) &= \text{entrop} \end{aligned} \tag{4.1}$$

This sets of work Schulman, Wolski, et al. 2017; Schulman, Levine, et al. 2015 considers limiting the update of the policy gradient at each step to guarantee *coherent* behavior.

4.1.3 Temporal Regularization

Though no work explicitely considers the bias and variance induced of considering the past estimates, several work attempts to exploit this relationship to better performance.

4.1.3.1 Natural value approximator

The closest work to ours is possibly Xu et al. 2017, where they define natural value approximator by projecting the previous states estimates by adjusting for the reward

and γ. Their formulation, while sharing similarity in motivation, yields different theory and algorithm in practice.

4.1.3.2 Backward bootstrapping

Backward bootstrapping method's can be seen as regularizing in feature space based on temporal proximity **sutton2009fast**; **li2008worst**; **baird1995residual**

4.1.4 Deliberation cost

There exists between temporal regularization and option. Indeed having a high deliberation cost Harb et al. 2017 means that you smooth out your problem by having longer action. This can be seen as a bias variance trade off.

4.1.4.1 Temporal coherence

Though it has been shown that exploiting *temporal coherence* can benefit to tracking algorithms and thus help in meta-learning R. S. Sutton, Koop, and Silver 2007, this remains far from the setting considered in this work.

5

Temporal Regularization

5.1 Value Based Temporal Regularization

Regularization in the feature/state space, or *spatial regularization* as we call it, exploits the regularities that exist in the observation (or state). In contrast, *temporal regularization* considers the temporal structure of the value estimates through a trajectory. Practically this is done by smoothing the value estimate of a state using estimates of states that occurred earlier in the trajectory. In this section we first introduce the concept of temporal regularization and discuss its properties in the policy evaluation setting. We then show how this concept can be extended to exploit information from the entire trajectory by casting temporal regularization as a time series prediction problem.

5.1.1 Definition

Let us focus on the simplest case where the value estimate at the current state is regularized using only the value estimate at the previous state in the trajectory, yielding

updates of the form

$$V_\beta(s_t) = \mathop{\mathbb{E}}_{s_{t+1},s_{t-1}\sim\pi}[r(s_t) + \gamma((1-\beta)V_\beta(s_{t+1}) + \beta V_\beta(s_{t-1}))]$$
$$= r(s_t) + \gamma(1-\beta)\sum_{s_{t+1}\in\mathcal{S}} p(s_{t+1}|s_t)V_\beta(s_{t+1}) + \gamma\beta\sum_{s_{t-1}\in\mathcal{S}} \frac{p(s_t|s_{t-1})p(s_{t-1})}{p(s_t)}V_\beta(s_{t-1}), \tag{5.1}$$

for a parameter $\beta \in [0,1]$. It can be rewritten in matrix form as $V_\beta = r + \gamma(((1-\beta)P^\pi + \beta\widetilde{P^\pi})V_\beta)$, where $\widetilde{P^\pi}$ corresponds to the reversal Markov chain of the MDP. We define a temporally regularized Bellman operator as:

$$\mathcal{T}_\beta^\pi v_\beta = r + \gamma((1-\beta)P^\pi v_\beta + \beta\widetilde{P^\pi}v_\beta). \tag{5.2}$$

To alleviate the notation, we denote P^π as P and $\widetilde{P^\pi}$ as $\widetilde{P}$.

Remark. *For $\beta = 0$, Eq. 5.2 corresponds to the original Bellman operator.*

We can prove that this operator has a unique fixed point, V_β^π.

Theorem 2. *The operator $\mathcal{T}_\beta^\pi$ has a unique fixed point V_β^π and $\mathcal{T}_\beta^\pi$ is a contraction mapping.*

Proof. We first prove that $\mathcal{T}_\beta^\pi$ is a contraction mapping in L_∞ norm. We have that

$$\begin{aligned}\left\|\mathcal{T}_\beta^\pi u - \mathcal{T}_\beta^\pi v\right\|_\infty &= \left\|r + \gamma((1-\beta)Pu + \beta\widetilde{P}u) - (r + \gamma((1-\beta)Pv + \beta\widetilde{P}v))\right\|_\infty \\ &= \gamma\left\|((1-\beta)P + \beta\widetilde{P})(u-v)\right\|_\infty \\ &\leq \gamma\left\|u-v\right\|_\infty,\end{aligned} \tag{5.3}$$

where the last inequality uses the fact that the convex combination of two row stochastic matrices is also row stochastic (he proof can be found in the appendix). Then using Banach fixed point theorem, we obtain that V_β^π is a unique fixed point. □

Algorithm 6 Policy evaluation with temporal regularization

Input: $\pi, \alpha, \gamma, \beta, \lambda$
$p = V(s)$
for all steps **do**
  Choose $a \sim \pi(\mathcal{S})$
  Take action a, observe $r(s), s'$
  $V(s) = r(s) + \gamma((1-\beta)V(s') + \beta p)$
  $p = V(s)$
end for

Furthermore the new induced Markov chain $(1-\beta)P+\beta\widetilde{P}$ has the same stationary distribution as the original P (the proof can be found in the appendix).

Lemma 3. *P and $(1-\beta)P+\beta\widetilde{P}$ have the same stationary distribution μ* $\quad \forall\beta \in [0,1]$.

5.1.2 Bias

In the policy evaluation setting, the bias between the original value function V^π and the regularized one V^π_β can be characterized as a function of the difference between P and its Markov reversal $\widetilde{P}$, weighted by β and the reward distribution.

Property 1. *Let $v^\pi = \sum_{i=0}^\infty \gamma^i P^i r$ and $v^\pi_\beta = \sum_{i=0}^\infty \gamma^i((1-\beta)P + \beta\widetilde{P})^i r$. We have that*

$$\begin{aligned} \left\|v^\pi - v^\pi_\beta\right\|_p &= \left\|\sum_{i=0}^\infty \gamma^i (P^i - ((1-\beta)P + \beta\widetilde{P})^i) r\right\|_p \\ &\leq \sum_{i=0}^\infty \gamma^i \left\|(P^i - ((1-\beta)P + \beta\widetilde{P})^i) r\right\|_p . \end{aligned} \tag{5.4}$$

This quantity is naturally bounded for $\gamma < 1$.

Remark. *Let P^∞ denote a matrix where columns consist of the stationary distribution μ. By the property of reversal Markov chains and lemma 3, we have that $\lim_{i\to\infty} \|P^i r - P^\infty r\| \to 0$ and $\lim_{i\to\infty} \|((1-\beta)P + \beta\widetilde{P})^i r - P^\infty r\| \to 0$, such that the Marvov chain P and its reversal $(1-\beta)P+\beta\widetilde{P}$ converge to the same value. Therefore, the norm $\|(P^i - ((1-\beta)P + \beta\widetilde{P})^i) r\|_p$ also converges to 0 in the limit.*

Remark. *It can be interesting to note that if the chain is reversible, meaning that* $P = \widetilde{P}$*, then the fixed point of both operators is the same, that is* $v^{\pi} = v^{\pi}_{\beta}$*.*

5.1.3 Variance

If the chain is reversible than I can find a factor of 2 on the variance because you just get twice more sample. In the non reversible case same thing. If you were to be able to sample from the newly generated P_β then you would solve it half as fast. Could be a cool argument.

5.1.4 Average reward case:

The temporally regularized MDP has the same average reward as the original one as it is possible to define the average reward Tsitsiklis and Van Roy 2002 as a function of the stationary distribution π, the reward vector and γ . This leads to the following property (the proof can be found in the appendix).

Property 2. *For a reward vector r, the MDPs defined by the the transition matrices* P *and* $(1-\beta)P + \beta\widetilde{P}$ *have the same average reward* ρ*.*

Intuitively this means that temporal regularization only reweigh the reward on each state based on the Markov reversal while preserving the average reward.

5.1.5 Temporal Regularization as a time series prediction problem:

It is possible to cast this problem of temporal regularization as a time series prediction problem and use richer models of temporal dependencies, such as exponential smoothing Gardner 2006, ARMA model Box, Jenkins, and Reinsel 1994, etc. We can write the update in a general form using n different regularizers $(\widetilde{v_0}, \widetilde{v_1}...\widetilde{v_{n-1}})$:

$$V(s_t) = r(s) + \gamma \sum_{i=0}^{N} [\beta(i)\widetilde{V}_i(s_{t+1})], \tag{5.5}$$

where $\widetilde{V}_0(s_{t+1}) = V(s_{t+1})$ and $\sum_{i=0}^{N} \beta(i) = 1$. For example, using exponential smoothing where $\widetilde{V}(s_{t+1}) = (1-\lambda)V(s_{t-1}) + (1-\lambda)\lambda V(s_{t-2})...$, the update can be written in operator form as:

$$\mathcal{T}_\beta^\pi v = r + \gamma\left((1-\beta)\, Pv + \beta\,(1-\lambda) \sum_{i=1}^{\infty} \lambda^{i-1} \widetilde{P}^i v \right), \tag{5.6}$$

and a similar argument as Theorem 1 can be used to show the contraction property.

Using more powerful regularizers could be beneficial, for example to reduce variance by smoothing over more values (exponential smoothing) or to model the trend of the value function through the trajectory (trend adjusted model). An example a temporal policy evaluation with temporal regularization using exponential smoothing can be found in the appendix.

Algorithm 7 Policy evaluation with exponential smoothing

```
Input: π, α, γ, β, λ
p = V(s)
for all steps do
    Choose a ∼ π(S)
    Take action a, observe r(s), s'
    V(s) = r(s) + γ((1 − β)V(s') + βp)
    p = (1 − λ)V(s) + λp
end for
```

5.1.6 Control

Temporal regularization can be extended to control by modifying the target of the value function (or the Q values) using temporal regularization. Experiments (Sec. 5.2.7) present an example of how temporal regularization can be applied within an actor-critic framework. The theoretical analysis of the control case is outside the scope of this paper.

5.1.7 Temporal difference with function approximation

It is also possible to extend temporal regularization using function approximation such as semi-gradient TD R. S. Sutton and Barto 2017. Assuming a function V_θ^β

parameterized by θ, we can consider $r(s) + \gamma((1-\beta)V_\theta^\beta(s_{t+1}) + \beta V_\theta^\beta(s_{t-1})) - V_\theta^\beta(s_t)$ as the target and differentiate with respect to $V_\theta^\beta(s_t)$. An example of a temporally regularized semi-gradient TD algorithm can be found in the appendix.

Algorithm 8 Temporally regularized semi-gradient TD

1: Input: policy π,β,γ
2: **for all** steps **do**
3: Choose $a \sim \pi(s_t)$
4: Take action a, observe $r(s), s_{t+1}$
5: $\theta = \theta + \alpha(r + \gamma((1-\beta)V_\theta(s_{t+1}) + \beta V_\theta(s_{t-1})) - V_\theta(s_t))\nabla V_\theta(s_t)$
6: **end for**

Property 3. *For a reward vector r, the MDP defined by the the transition matrix P and* $(1-\beta)P + \beta\widetilde{P}$ *have the same average reward* ρ.

$$\frac{\rho}{1-\alpha} = \sum_i^\infty \gamma^i \pi^T r. \tag{5.7}$$

Lemma 4. *P and* $(1-\beta)P+\beta\widetilde{P}$ *have the same stationary distribution* $\mu \quad \forall\beta \in [0,1]$.

Proof. It is known that P^π and $\widetilde{P^\pi}$ have the same stationary distribution. Using this fact we have that

$$\begin{aligned}\mu((1-\beta)P^\pi + \beta\widetilde{P^\pi}) &= (1-\beta)\mu P^\pi + \beta\mu\widetilde{P^\pi} \\ &= (1-\beta)\mu + \beta\mu \\ &= \mu.\end{aligned} \tag{5.8}$$

□

Lemma 5. *The convex combination of two row stochastic matrices is also row stochastic.*

Proof. Let e be vector a columns vectors of 1.

$$\begin{aligned}(\beta P^\pi + (1-\beta)\widetilde{P^\pi})e &= \beta P^\pi e + (1-\beta)\widetilde{P^\pi}e \\ &= \beta e + (1-\beta)e \\ &= e.\end{aligned} \tag{5.9}$$

□

5.1.8 Connection with multi-step methods

5.2 Experiments

We now presents empirical results illustrating potential advantages of temporal regularization, and characterizing its bias and variance effects on value estimation and control.

5.2.1 Mixing time

This first experiment showcases that the underlying Markov chain of a MDP can have a smaller mixing time when temporally regularized. The mixing time can be seen as the number of time steps required for the Markov chain to get *close enough* to its stationary distribution. Therefore, the mixing time also determines the rate at which policy evaluation will converge to the optimal value function Baxter and Bartlett 2001. We consider a synthetic MDP with 10 states where transition probabilities are sampled from the uniform distribution. Let P^{∞} denote a matrix where columns consists of the stationary distribution μ. To compare the mixing time, we evaluate the error corresponding to the distance of P^i and $\left((1-\beta)P+\beta\widetilde{P}\right)^i$ to the convergence point P^{∞} after i iterations. Figure 5.1 displays the error curve when varying the regularization parameter β. We observe a U-shaped error curve, that intermediate values of β in this example yields faster mixing time. One explanation is that transition matrices with extreme probabilities (low or high) yield poorly conditioned transition matrices. Regularizing with the reversal Markov chain often leads to a better conditioned matrix at the cost of injecting bias.

5.2.2 Bias

It is well known that reducing variance comes at the expense of inducing (smaller) bias. This has been characterized previously (Sec. 5.1) in terms of the difference

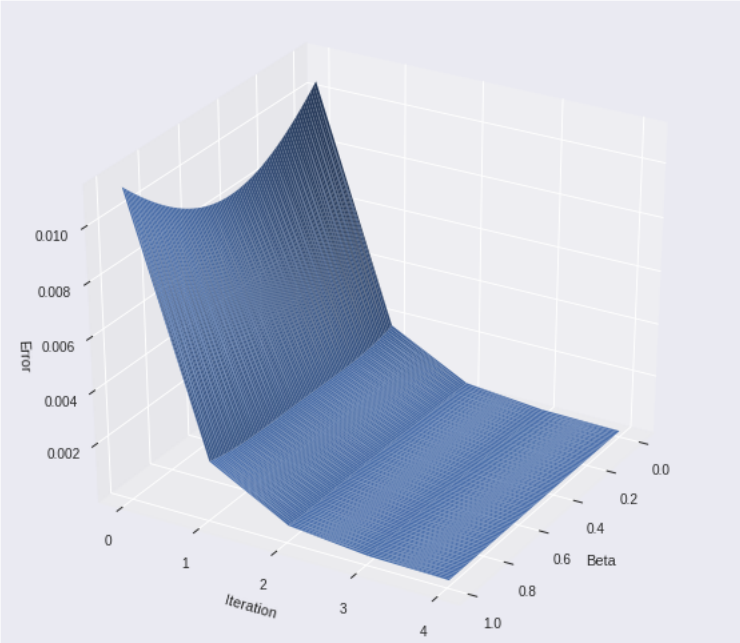


Figure 5.1: Distance between the stationary transition probabilities and the estimated transition probability for different values of regularization parameter β.

between the original Markov chain and the reversal weighted by the reward. In this experiment, we attempt to give an intuitive idea of what this means. More specifically, we would expect the bias to be small if values along the trajectories have similar values. To this end, we consider a synthetic MDP with 10 states where both transition functions and rewards are sampled randomly from a uniform distribution. In order to create temporal dependencies in the trajectory, we smooth the rewards of N states that are temporally close (in terms of trajectory) using the following formula: $r(s_t) = \frac{r(s_t)+r(s_{t+1})}{2}$. Figure 5.2 shows the difference between the regularized and unregularized MDPs as N changes, for different values of regularization parameter β. We observe that increasing N, meaning more states get rewards close to one another, results into less bias. This is due to rewards putting emphasis on states where the original and reversal Markov chain are similar.

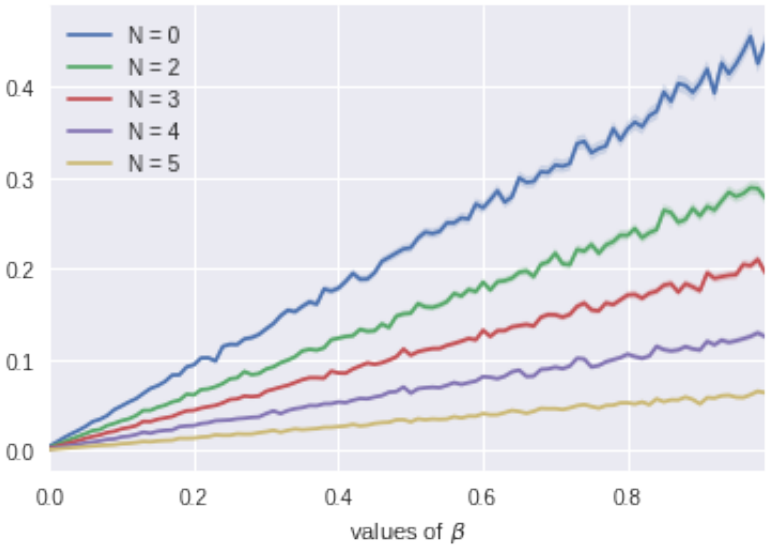


Figure 5.2: Mean difference between V_β^π and V^π given the regularization parameter β, for different amount of smoothed states N.

5.2.3 Variance number 2

You could use a random walk. The P is reversible to so no bias. Then compare the performance of monte carlo, TD and our thing. Should just be faster. In this case you want to compare with the number of samples used and try to show an increase of 2 folds. its like having double the sample link the variance from theory before

5.2.4 Variance

The primary motivation of this work is to reduce variance, therefore we now consider an experiment targeting this aspect. Figure 5.3 shows an example of a synthetic, three states MDP, where the variance of S_1 is (relatively) high. We consider an agent that is evolving in this world, changing states following the stochastic policy indicated. We are interested in the error when estimating the optimal state value of S_1, $V^*(S_1)$, with and without temporal regularization, denoted $V_\beta^\pi(S_1)$, $V^\pi(S_1)$, respectively.

Figure 5.4 shows these errors at each iteration, averaged over 100 runs. We observe that temporal regularization indeed reduces the variance and thus helps the learning process by making the value function easier to learn.

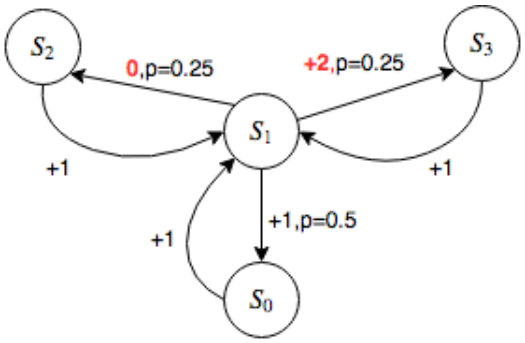


Figure 5.3: Synthetic MDP where state S_1 has high variance.

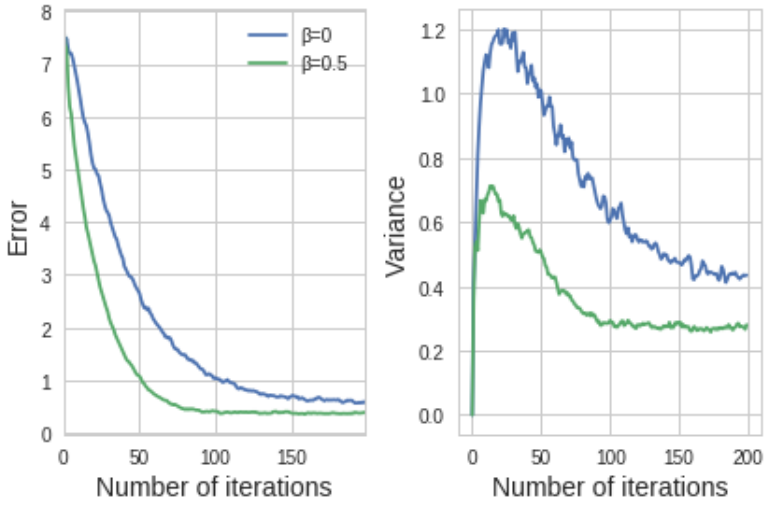


Figure 5.4: Left plot showsabsolute difference between original ($V^{\pi}(S_1)$) and regularized ($V^{\pi}_{\beta}(S_1)$) state value estimates to the optimal value $V^{*}(S_1)$. Right plot shows the variance of the estimates V.

5.2.5 Propagation of the information

We now illustrate with a simple experiment how temporal regularization allows the information to spread faster among the different states of the MDP. For this purpose, we consider a simple MDP, where an agent walks randomly in two rooms (18 states) using four actions (up, down, left, right), and a discount factor $\gamma = 0.9$. The reward is $r_t = 1$ everywhere and passing the door between rooms (shown in red on Figure 5.5) only happens 50% of the time (on attempt). The episode starts at the top left and terminates when the agent reaches the bottom right corner. The sole goal is to learn the optimal value function by walking along this MDP (this is not a race toward the

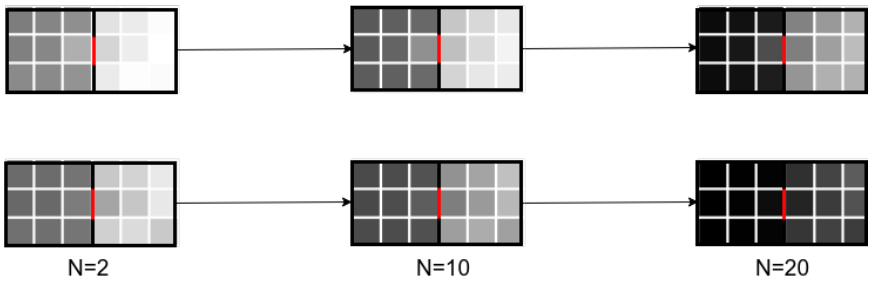


Figure 5.5: Proximity of the estimated state value to the optimal value after N trajectories. Top row is the original room environment and bottom row is the regularized one ($\beta = 0.5$). Darker is better.

end).

Figure 5.5 shows the proximity of the estimated state value to the optimal value with and without temporal regularization. The darker the state, the closer it is to its optimal value. The heatmap scale has been adjusted at each trajectory to observe the difference between both methods. We first notice that the overall propagation of the information in the regularized MDP is faster than in the original one. We also observe that, when first entering the second room, bootstrapping on values coming from the first room allows the agent to learn the optimal value faster. This suggest that temporal regularization could help agents explore faster by using their prior from the previous visited state for learning the corresponding optimal value faster.

5.2.6 Noisy state representation

The next experiment illustrates a major strength of temporal regularization, that is its robustness to noise in the state representation. This situation can naturally arise when the state sensors are noisy or insufficient to avoid aliasing. For this task, we consider the synthetic, one dimensional, continuous setting displayed in Figure 5.6. A learner evolving in this environment walks randomly along this line with a discount factor $\gamma = 0.5$. Let $x_t \in [0, 1]$ denote the position of the agent along the line at time t. The next position $x_{t+1} = x_t + a_t$, where action $a_t \sim \mathcal{N}(0, 0.1)$. The state of the agent corresponds to the position perturbed by a zero-centered Gaussian noise ϵ_t,

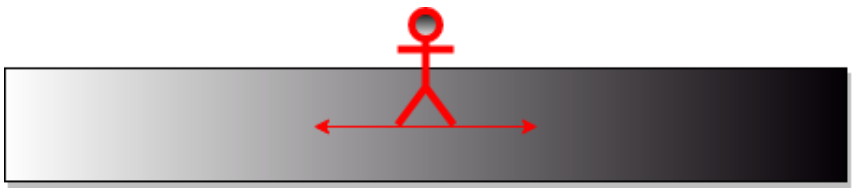


Figure 5.6: Synthetic, one dimensional, MDP where the reward corresponds to the position of the agent in $[0, 1]$.

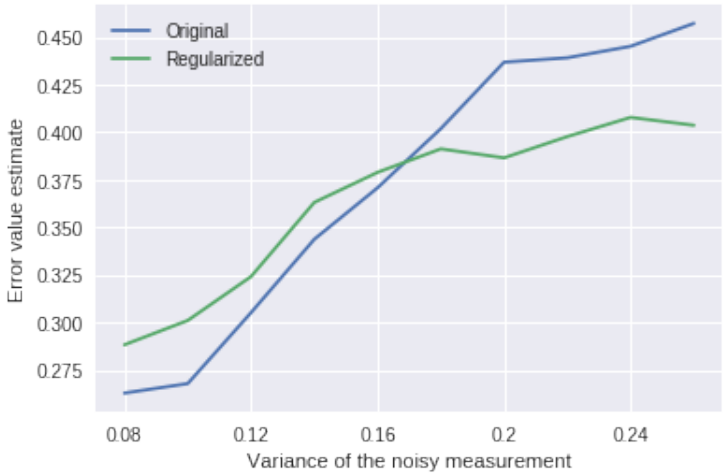


Figure 5.7: Absolute distance from the original (θ^{π}) and the regularized (θ^{π}_{β}) state value estimates to the optimal parameter θ^* given the noise variance σ^2 in state sensors.

such that $s_t = x_t + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ are i.i.d. When the agent moves to a new position x_{t+1}, it receives a reward $r_t = x_{t+1}$. The episode ends after 100 steps. In this experiment we model the value function using a linear model with a single parameter θ. We are interested in the error when estimating the optimal parameter function θ^* with and without temporal regularization, that is θ^{π}_{β} and θ^{π}, respectively. In this case we use the TD version of temporal regularization presented at the end of Sec. 5.1. Figure 5.7 shows these errors, averaged over 100 repetitions, for different values of noise variance σ^2. We observe that as the noise variance increases, the un-regularized estimate becomes less accurate, while temporal regularization is more robust. This experiment highlights a case where temporal regularization is effective even in the absence of smoothness in the state space (which other regularization methods would target). This is further highlighted in the next experiments.

5.2.7 Deep reinforcement learning

To showcase the potential of temporal regularization in high dimensional settings, we adapt an actor-critic based method (PPO Schulman, Wolski, et al. 2017) using temporal regularization. More specifically, we incorporate temporal regularization as exponential smoothing in the target of the critic. We evaluate the performance on multiple Atari games, where we consider the following performance measure Xu et al. 2017:

$$\frac{\text{regularized} - \text{baseline}}{\text{baseline} - \text{random}}. \tag{5.10}$$

The hyper-parameters for the temporal regularization are $\beta = 0.2$ and a decay of $1e^{-5}$. Those are selected on 7 games and 3 training seeds. All other hyper-parameters correspond to the one used in the PPO paper. Our implementation[1] is based on the publicly available OpenAI codebase Dhariwal et al. 2017. The previous four frames are considered as the state representation Mnih, Kavukcuoglu, Silver, Rusu, et al. 2015. For each game, 10 independent runs (10 random seeds) are performed and the performance are reported in Figure 5.8.

Figure 5.8 shows that adding temporal regularization improves the performance on multiple games. This suggests that the regularized optimal value function may be smoother and thus easier to learn, even when using function approximation with deep learning. Also, as shown in previous experiments (Sec. 5.2.6), temporal regularization being independent of spatial representation makes it more robust to mis-specification of the state features, which is a challenge in some of these games (e.g. when assuming full state representation using some previous frames).

[1]The code can be found in the supplementary material.

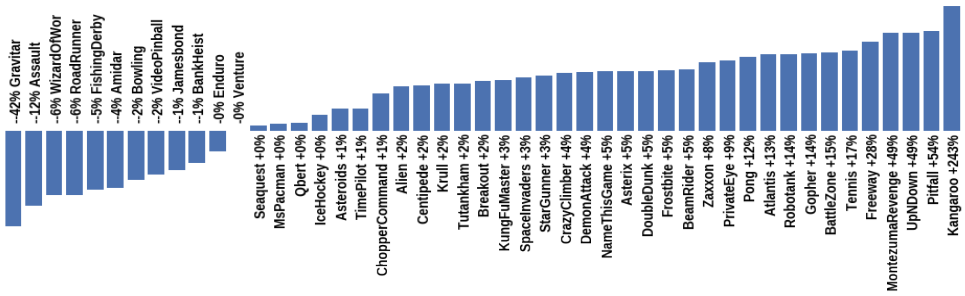


Figure 5.8: Performance (Eq. 5.10) of a temporally regularized PPO on a suite of Atari games.

5.3 Policy Gradient Temporal Regularization

This idea is a continuation of the paper I wrote recently about what I call "Temporal Regularization". The main intuition is to exploit previous estimate (v, q or pi's) along the trajectory to reduce the variance of the current estimate. In particular in this project we are interested in policy gradient. We want to regularize the objective function by constraining the policy to be more "consistent" temporally

$$R(\pi) = \beta(\pi(a_{t-1}|s_{t-1}) - \pi(a_t|s_t)) \tag{5.11}$$

It is possible to see this formulation as some kind of entropy regularization where the regularization is done in trajectory space compared to PPO ect... where it is done in policy space. Another perspective is to view π as a time series and apply some entropy regularization to the prediction of the time series. Some of the advantages of doing this kind of regularization could be more consistent behavior in time, exploration, variance reduction..

Theoretically we want to properly define this entropy using concepts of dynamical system and prove its potential convergence/convexity following the proof's used Neu, Jonsson, and Gómez 2017.

More particularly in dynamical system there are several ways to define entropy. The

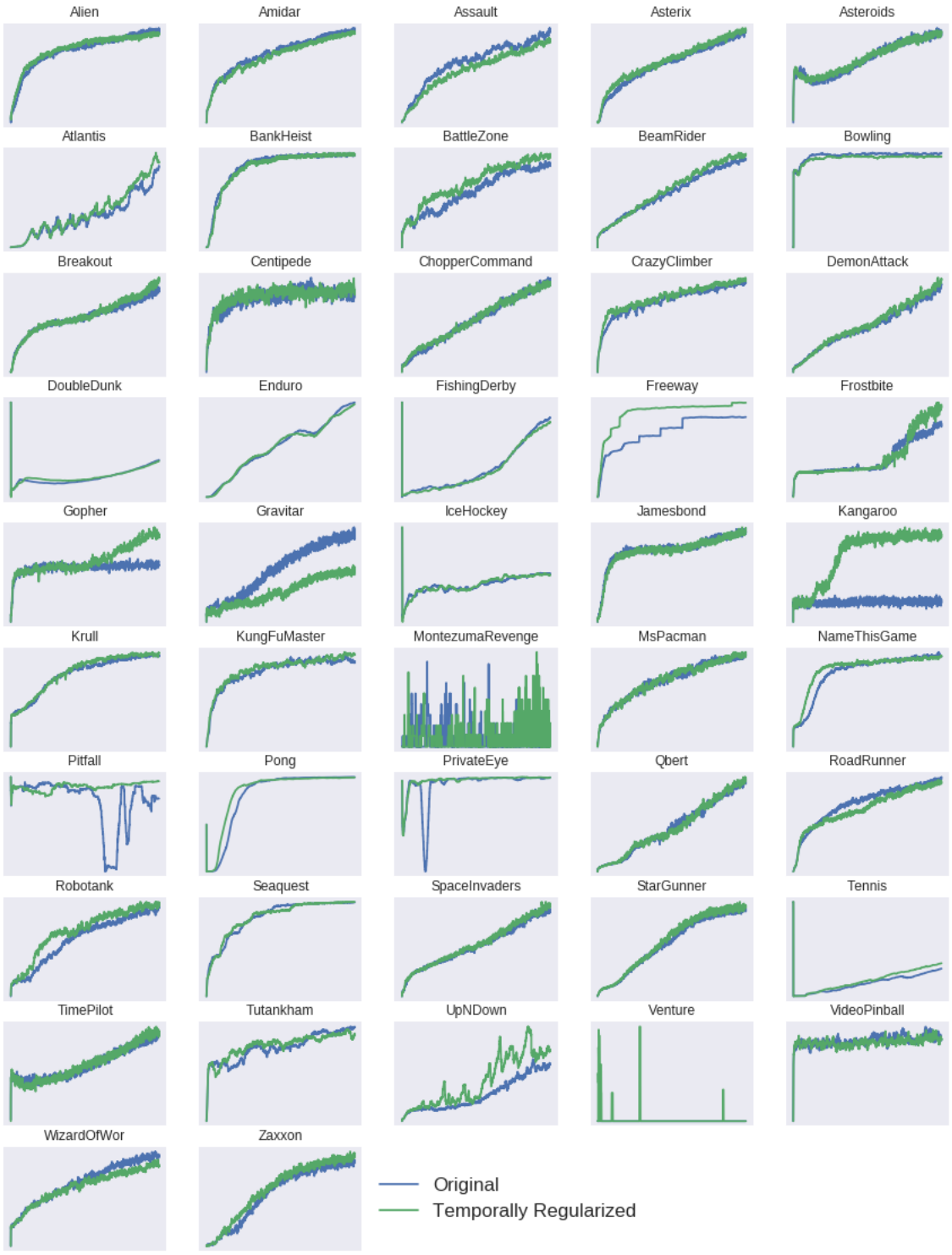


Figure 5.9: Average reward per episode on Atari games.

community has not agreed on one in particularly form. Many of them are based on the kolmogorov sinai entropy. Intuitively they often define the entropy over each partition of your dynamical system and take a sup over that. In our case depending on how we average over the past we would implicitely define our partition function. This reference http://prac.im.pwr.edu.pl/~downar/english/documents/entropy.pdf is a good summary of the different form of entropy used. In particular I believe we can use something similar to the dynamical entropy explained in section 3.4.

Intuitively we want to minimize the "temporal entropy" when you are in an option/state of your dynamical system to get coherent behavior and maximize it when you change state/option. It is very related to option and deliberation cost Harb et al. 2017 as they minimize the entropy in an option by adding a cost to changing option. I believe deliberation cost can actually be cast as some kind of entropy minimization where your option defines the partition function. In our case by simply taking a combination of the previous step(or more) it is kind like implicitly defining an option with states that are around you.

6

Recurrent Learning

In the previous chapter we regularized the value estimate by adding a regularization term to the objective(target). In this chapter we explore explicitly averaging the value estimate using exponential smoothing.

Résumé

In sequential modelling, exponential smoothing is one of the most widely used techniques to maintain temporal consistency in estimates. In this work, we propose Recurrent Learning, a method that estimates the value function in reinforcement learning using exponential smoothing *along the trajectory*. We establish its asymptotic convergence properties under some smoothness assumption on the reward. The proposed algorithm yields a natural way to learn a state dependent emphasis function that selectively learns to emphasize or ignore states based on trajectory information. We demonstrate the potential for this selective updating on a partially observable domain and several continuous control tasks.

6.1 Introduction

Reinforcement Learning is a mathematical framework designed to model sequential decision making. It is used to solve a wide range of control tasks ranging from video

games Vinyals et al. 2017; Mnih, Kavukcuoglu, Silver, Graves, et al. 2013; Mnih, Badia, et al. 2016 to robotics Kober, Bagnell, and Peters 2013; Abbeel, Coates, and Ng 2010. It is also used in many real-life applications such as hydro control Grinberg, Precup, and Gendreau 2014 and power grid management François-Lavet et al. 2016.

However, reinforcement learning algorithms suffer from several issues. In particular, we describe two issues that Recurrent Learning attempts to mitigate. The first one considers the *variance of the value estimates through the trajectory*. Consider a hypothetical scenario in which a person is driving a car at a high speed. Its behavior needs to be temporally coherent. If the driver wants to change lane, there needs to be a continuous smooth decision making process. For the policy of the driver to be coherent, its value function needs to be temporally smooth. Most algorithms make a decision at every time step without necessarily *explicitly* enforcing temporal coherence nor considering previous decisions. This can lead to erratic and temporally inconsistent behaviors, particularly in tabular and discrete settings. The second issue considers the limited capacity of the brain to process information and make decision. Bounded rationality argues that human brain has a limited capacity to learn and can't store all the information. In this work we argue that the capacity to ignore or emphasize chosen state is a key component for the success of decision making algorithms. To summarize, this work proposes a method that attempts to address the following two aspects. 1) Minimization of the variance of the estimates through the *trajectory*. 2) Learn a mechanism that emphasizes and de-emphasizes updates on relevant states. Exponential smoothing Gardner Jr 1985 is one of the most widely used technique to reduce the variance of point estimates using *previous observations*.

$$x_t = \beta s_t + (1 - \beta) x_{t-1} \tag{6.1}$$

where s_t is a signal, x_t the averaged signal at time step t and β a smoothing factor. In time series data it is used to approximate the mean of a stream of data. This kind of smoothing has various names Polyak and Juditsky 1992; Kingma and Ba 2014

depending on the field it is used in. Most quantities in reinforcement learning (Q values, state values, actions, etc) are estimated as point estimate ignoring previous estimates from the trajectory. We propose to use exponential smoothing in reinforcement learning on the trajectory. In other words, we propose to smooth the estimate of a state s_t using the estimates of states s_i that occur earlier in the trajectory ($i \geq t$). Intuitively, states that are temporally close to each other should have similar value. However, exponential averaging can be biased if a sharp change(non-stationarity) is encountered in the trajectory, like falling off a cliff. To alleviate this issue a common technique used is to set β_t the exponential smoothing factor as a state or time dependent. In Recurrent Neural Networks, it is possible to view LSTM Hochreiter and Schmidhuber 1997 and GRU J. Chung et al. 2014 as several non linear exponential smoothing functions. The key ingredient is the gating mechanism(state dependent β_t) used to update the hidden cell. The capacity to ignore information allows the cell to focus only on important information. This can also be found in semi-iterative method used to solve linear system of equations. Golub and Varga 1961 finds the optimal β_t using Chebyshev polynomials. In this work we explore a new method that attempts to learn a state dependent smoothing factor β. We show how it relates to existing methods and exploits similar properties such as emphatic TD R. S. Sutton, Mahmood, and M. White 2016; Mahmood et al. 2015. The contributions of the paper are as follows:

- Propose a new way to estimate value function in reinforcement learning by exploiting the estimates along the trajectory.
- Provide asymptotic convergence guarantee in the tabular setting under some smoothness assumption.
- Derive a learning rule for a state dependent β.
- Perform a set of experiments in tabular and continuous settings to evaluate its strengths and weaknesses.

6.2 Technical Background

A Markov Decision Process (MDP), as defined in Puterman 1994, consists of a discrete set of states $\mathcal{S}$, a transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$, and a reward function $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. On each round t, the learner observes current state $s_t \in \mathcal{S}$ and selects action $a_t \in \mathcal{A}$, after which it receives reward $r_t = r(s_t, a_t)$ and moves to a new state $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$. We define a stationary policy π as a probability distribution over actions conditioned on states $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$, such that $a_t \sim \pi(\cdot|s_t)$. When performing policy evaluation, the goal is to find the optimal value function V^π that estimates the discounted expected return of policy π at a state $s \in \mathcal{S}$, $V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s]$, with discount factor $\gamma \in [0, 1)$. In this paper we only consider policy evaluation and simplify the notation: $r(s_t) = r(s_t, a_t)$.

In practice V^π is approximated using Monte Carlo rollouts R. S. Sutton and Barto n.d. or TD methods R. S. Sutton 1988. In reinforcement learning the aim is to find a function $V_\theta : \mathbb{S} \to \mathbb{R}$ parametrized by θ that approximates V^π. We can fall back to the tabular setting by representing the states in a one hot vector form with $\theta \in \mathbb{R}^{|\mathbb{S}|}$. The goal is to find a set of parameters θ that minimizes the squared loss:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{\pi}[(V^\pi - V_\theta)^2] \tag{6.2}$$

which yields the following update by taking the derivative with respect to θ:

$$\theta_{t+1} = \theta_t + \alpha(V^\pi(s_t) - V_{\theta_t}(s_t))\nabla_{\theta_t} V_\theta(s_t) \tag{6.3}$$

where α is a learning rate.

In the tabular setting, convergence is usually proven by casting the learning algorithm as a stochastic approximation Tsitsiklis 1994; Borkar 2009; Borkar and Meyn 2000 of the form:

$$\theta_{t+1} = \theta_t + \alpha(\mathcal{T}\theta_t - \theta_t + w(t)) \tag{6.4}$$

where $\mathcal{T} : \mathbb{R}^{|\mathbb{S}|} \to \mathbb{R}^{|\mathbb{S}|}$ is a contraction operator and $w(t)$ is a noise term. As an example, TD(0) is known to converge to the fixed point of the bellman operator R. S.

Sutton 1988:

$$\mathcal{T}V_\theta(s_t) = \mathop{\mathbb{E}}_{s_{t+1}\sim\pi}[r(s_t) + \gamma V_\theta(s_{t+1})] \tag{6.5}$$

However, in practice we have access to a noisy version of the operator $\widetilde{\mathcal{T}}$ due to sampling process hence the noise term $w(t)$:

$$w(t) = r_t + \gamma V_\theta(s_{t+1}) - \mathop{\mathbb{E}}_{s_{t+1}\sim\pi}[r + \gamma V_\theta(s_{t+1})] \tag{6.6}$$

6.3 Recurrent Learning in Reinforcement Learning

In this work, we propose to estimate the value function using the estimates along the trajectory. The value of a state s at time step t is given by:

$$V_\theta^\beta(s_t) = \beta_t V_\theta(s_t) + (1-\beta_t)(V_\theta^\beta(s_{t-1})) \tag{6.7}$$

where V_θ^β is an exponential smoothing of V_θ. V_θ^β is a function parametrized by θ and β. We consider β to be fixed for each state in this section and relax the assumption later. For a given set of state dependant $\beta(s_t) = \beta_t$, the goal is to find a set of parameters θ minimizing,

$$\min_\theta \mathcal{L}(\theta) = \min_\theta \mathop{\mathbb{E}}_{\pi}[(V^\pi - V_\theta^\beta)^2] \tag{6.8}$$

In contrast to traditional methods that attempts to minimize Eq. 6.2 we explicitly enforce temporal consistency by finding θ that minimizes Eq. 6.8. By taking the derivative with respect to Eq. 6.8 we obtain the following update:

$$\theta = \theta + \alpha\delta_t \nabla_\theta V_\theta^\beta(s_t) \tag{6.9}$$

where $\delta_t = V^\pi(s_t) - V_\theta^\beta(s_t)$. The gradients of V_θ^β can be expressed in a recursive form:

$$\nabla_\theta V_\theta^\beta(s_t) = \beta_t \nabla_\theta V_\theta + (1-\beta_t)\nabla_\theta V_\theta^\beta(s_{t-1}) \tag{6.10}$$

For computational reason, it is possible to approximate the gradient $\nabla_\theta V_\theta^\beta(s_t)$ using a recursive *eligibility* trace:

$$e_t = \beta_t \nabla_\theta V_\theta(s_t) + (1 - \beta_t) e_{t-1} \tag{6.11}$$

The potential bias induced by this approximation are discussed later in the paper. We present Recurrent Temporal Difference(RTD) in Algorithm 9.

Algorithm 9 Recurrent Temporal Difference, RTD(0)

Input: π,β,γ,θ
Initialize: $V_\theta^\beta(s_0) = V_\theta(s_0)$ and $e_0 = \nabla_\theta V_\theta(s_0)$
Output: θ
for all t **do**
  Choose $a \sim \pi(s_t)$
  Take action a, observe $r(s_t), s_{t+1}$
  $V_\theta^\beta(s_t) = \beta_t V_\theta(s_t) + (1 - \beta_t)(V_\theta^\beta(s_{t-1})$
  $e_t = \beta_t \nabla_\theta V_\theta(s_t) + (1 - \beta_t) e_{t-1}$
  $\delta_t = r(s_t) + \gamma V_\theta(s_{t+1}) - V_\theta^\beta(s_t)$
  $\theta = \theta + \alpha \delta_t e_t$
end for

In practice, $V_\theta^\beta(s_0)$ is initialized with the value of the first state $V_\theta(s_0)$. We can understand our algorithm better when we intuitively interpret its effect in the *extreme cases* ($\beta = \{0, 1\}$). We consider TD(0) in a tabular setting for simplicity. First, consider a state s_t where $\beta(s_t) = 0$. The value of this state is frozen at the initialization point and is never updated. This is because the trace as defined in Eq. (6.11) is $e_t = e_{t-1}$ making such a state *ineligible* for the update. The error received at this state is used to update the previous states as per their *eligibility* in e_{t-1}. This means that for a state s_t with $\beta(s_t) = 1$, $V_\theta(s_t)$ is updated at every time step $t + n$ until another state with $\beta(s_{t+n}) = 1$ is encountered.

For any $i \leq t$ where $\beta_i \in (0, 1]$, it is possible to express $V_\theta^\beta(s_t)$ as:

$$\begin{aligned} V_\theta^\beta(s_t) &= V_\theta(s_i) - \Delta_t(s_i) \\ \Delta_t(s_i) &= (1 - C_t(s_i))(V_\theta(s_i) - \widetilde{V}_t(s_i)) \end{aligned} \tag{6.12}$$

where $C_t(s_i) = \beta_i \prod_{p=i+1}^{t}(1 - \beta_p)$, $\widetilde{V}_t(s_t)$ is a convex combination of all V_θ encountered in the trajectory except $V_\theta(s_i)$ weighted by their respective contribution(β) to the

estimate $V_\theta^\beta(s_t)$. For example, if we consider updating $V_\theta(s_2)$ at $t = 3$ and have the following $\beta_1 = 0.9, \beta_2 = 0.1, \beta_3 = 0.1$. The value of $\widetilde{V}_3(s_2)$ will be mainly composed of $V_\theta(s_1)$. The main component of the error will be $V^\pi(s_3) - V_\theta(s_1)$. We take an example with $t = 3$ and consider $i = 2$:

$$\begin{aligned} V_\theta^\beta(s_3) &= \beta_3 V(s_3) + (1-\beta_3)\beta_2 V(s_2) + (1-\beta_3)(1-\beta_2)V(s_1) \\ &= V(s_2) - (1-(1-\beta_3)\beta_2)(V(s_2) - \frac{\beta_3 V(s_3) + (1-\beta_3)(1-\beta_2)V(s_1)}{(1-(1-\beta_3)\beta_2)}) \\ &= V(s_2) - (1-(1-\beta_3)\beta_2)(V(s_2) - \widetilde{V}_t(s_i)) \end{aligned} \tag{6.13}$$

To see that $\widetilde{V}$ is a convex combination of the all the V encountered along the trajectory weight by β except $V(s_2)$ it suffices to see that:

$$\begin{aligned} &\frac{\beta_3 + (1-\beta_3)(1-\beta_2)}{(1-(1-\beta_3)\beta_2)} = 1 \\ &\equiv \beta_3 + (1-\beta_3)(1-\beta_2) = (1-(1-\beta_3)\beta_2) \\ &\equiv \beta_3 + (1-\beta_3)\beta_2 + (1-\beta_3)(1-\beta_2) = 1 \end{aligned} \tag{6.14}$$

where the last line is true because $\beta \in (0, 1]$Every update to $V_\theta(s_i)$ made during the trajectory at time step t can be decomposed as follows:

$$V_\theta(s_i) = V_\theta(s_i) + \alpha e_t(s_i)(V^\pi(s_t) + \Delta_t(s_i) - V_\theta(s_i)) \tag{6.15}$$

$\Delta_t(s_i)$ can be interpreted as a regularization term composed of the difference between $V_\theta(s_i)$ and $\widetilde{V}_t(s_i)$. In practice, one can observe an increase in the magnitude of this term with a decrease in *eligibility*. This suggests that the biased updates contribute less in the learning. Bounding the magnitude of Δ to ensure contraction is the key concept used in this paper to ensure asymptotic convergence.

6.3.1 Asymptotic convergence

Consider a tabular setting with TD(0). The main idea is to cast recurrent learning as an asynchronous stochastic approximation Tsitsiklis 1994 with an additional regularization term. By bounding the magnitude of this term we show that the operator

is a contraction. The algorithm is asynchronous because the eligibility trace updates only certain states at each time step.

Following the decomposition in Eq. 6.15 we can recover the stochastic approximation formulation described in Eq. 6.4 with the following operator $\mathcal{T}^{\beta} : \mathbb{R}^{|\mathbb{S}|} \rightarrow \mathbb{R}^{|\mathbb{S}|}$:

$$\mathcal{T}^{\beta} V(s_i) = \mathop{\mathbb{E}}_{\pi}[r_t + \gamma V_{\theta}(s_{t+1}) + \Delta_t(s_i)] \tag{6.16}$$

for all states s_i with $\beta_i \in (0, 1]$. We consider the following assumptions to prove convergence. The first assumption deals with the ergodic nature of the Markov chain. It is a common assumption in theoretical reinforcement learning that guarantees infinitely many visits to all states thereby avoiding chains with transient states.

Assumption 2. *The Markov chain is ergodic.*

The second assumption concerns the relative magnitude of the maximum and the minimum reward. It is a key element that allow us to bound the magnitude of the regularization term.

Assumption 3. *We define $R_{\max}$ and $R_{\min}$ as the maximum and minimum reward in a MDP. The relative magnitude between them is bounded by a factor of D such that:*

$$\begin{aligned} DR_{max} &\leq R_{min} \\ D &> \gamma \\ R_{min} &\geq 0 \end{aligned} \tag{6.17}$$

where $D \in (0.5, 1]$ is a constant to be defined based on γ.

This is the most restrictive assumption of the proof. Relaxation to this assumption is proposed in the discussions. Although assumption 3 is restrictive in theory we never observed a divergence in tabular and deep RL settings that we considered.

As mentioned earlier, the key component of the proof is to control the magnitude of the term in Eq. 6.15: $\Delta_t(s_i) = (1 - C_t(s_i))(V_{\theta}(s_i) - \widetilde{V}_t(s_i))$. As the eligibility of

this update gets smaller the magnitude of the term gets bigger. This suggests that not updating certain states whose eligibility is less than the threshold C can help mitigate biased updates. Depending on the values of γ and D we may need to set a threshold C to guarantee convergence.

Theorem 3. *Define* $V_{\max} = \frac{R_{\max}}{1-(\gamma+(1-D))}$ *and* $V_{\min} = \frac{R_{\min}}{1-(\gamma-(1-D))}$. $\mathcal{T}^{\beta} : X \to X$ *is a contraction operator if the following holds:*

- *Let* X *be the set of* V_θ *functions such that* $\forall s \in \mathbb{S} \quad V_{\min} \leq V_\theta(s) \leq V_{\max}$. *The functions* V_θ *are initialized in* X.
- *For a given* D *and* γ *we select* C *such that* $\Delta \leq (1-C)(V_{\max}-V_{\min}) \leq (1-D)V_{\min}$

Proof. The first step is to prove that $\mathcal{T}^{\beta}$ maps to itself for any noisy update $\widetilde{\mathcal{T}^{\beta}}$. From 2) we know that $(1-C)V_{\max} - V_{\min} < DV_{\min} \leq DV_{\max}$ we can then deduce that

$$\begin{aligned}\widetilde{\mathcal{T}^{\beta}}V(s) &\leq R_{\max} + \gamma V_{\max} + (1-C)(V_{\max} - V_{\min}) \\ &\leq R_{\max} + (\gamma + (1-D))V_{\max} \\ &\leq V_{\max}\end{aligned} \tag{6.18}$$

and

$$\begin{aligned}\widetilde{\mathcal{T}}^{\beta}V(s) &\geq R_{\min} + \gamma V_{\min} + (1-C)(V_{\min} - V_{\max}) \\ &\geq R_{\min} + (\gamma - (1-D))V_{\min} \\ &\geq V_{min}\end{aligned} \tag{6.19}$$

The next step is to show that $\mathcal{T}^{\beta}$ is a contractive operator:

$$\begin{aligned}&\left\|\mathcal{T}^{\beta}V - \mathcal{T}^{\beta}U\right\|_{\infty} \\ &\leq \max_{s,s'} \mathop{\mathbb{E}}_{\pi}[\gamma V(s) + \Delta^V(s') - (\gamma U(s) + \Delta^U(s'))] \\ &\leq \max_{s,s'} \mathop{\mathbb{E}}_{\pi}[\gamma(V(s) - U(s)) + (1-D)(V(s') - U(s'))] \\ &\leq \max_{s} \mathop{\mathbb{E}}_{\pi}[((1-D)+\gamma)(V(s) - U(s))] \\ &\leq ((1-D)+\gamma)\left\|V - U\right\|_{\infty}\end{aligned} \tag{6.20}$$

and from the assumption we know that $(1-D)+\gamma < 1$. □

We provide an example to decide a choice on C based on γ and D. To select C based on γ and D it suffice to solve analytically for:

$$\begin{aligned}
&(1-C)(V_{\max}-V_{\min}) \leq (1-D)V_{\min} \\
&\equiv (1-C)\frac{R_{\max}}{1-(\gamma+(1-D)} \leq ((1-D)+(1-C))\frac{R_{\min}}{1-(\gamma-(1-D)} \\
&\equiv \frac{(1-C)(1-(\gamma-(1-D)))}{(1-(\gamma+(1-D))((1-D)+(1-C)}R_{\max} \leq R_{\min} \\
&\equiv \frac{D(1-C)(1-(\gamma-(1-D)))}{(1-(\gamma+(1-D))((1-D)+(1-C)}R_{\min} \leq R_{\min}
\end{aligned} \tag{6.21}$$

which is satisfied only if:

$$\frac{D(1-C)(1-(\gamma-(1-D)))}{(1-(\gamma+(1-D))((1-D)+(1-C)} \leq 1 \tag{6.22}$$

As an example for $D = 0.8$ and $\gamma = 0.5$ any $C \geq 0.33$ satisfies this inequality.

We can guarantee that V_θ converges to a fixed point of the operator $\mathcal{T}^\beta$ with probability one using theorem 3 of Tsitsiklis 1994. We now discuss the assumptions of theorem 3 in Tsitsiklis 1994

Assumption 1: Allows for delayed update that can happen in distributed system for example. In this algorithm all V's are updated at each time step t and is not an issue here.

Assumption 2: As described by Tsitsiklis 1994 assumption 2 "allows for the possibility of deciding whether to update a particular component x_i at time t, based on the past history of the process.". This assumption is defined to accommodate for ϵ-greedy exploration in Q-learning. In this paper we only consider policy evaluation hence this assumptions holds.

Assumption 3: The learning rate of each state $s \in \mathbb{S}$ must satisfy Robbins Monroe conditions such that there exists $C \in \mathbb{R}$:

$$\begin{aligned}
\sum_{i=0}^{\infty} \alpha_t(s)e_t(s) = \infty \quad \text{w.p.1} \\
\sum_{i=0}^{\infty} (\alpha_t(s)e_t(s))^2 \leq C
\end{aligned} \tag{6.23}$$

This can be verified by assuming that each state gets visited infinitely often and an appropriate decaying learning rate based on $\#_s$ (state visitation count) is used (linear for example).

Assumption 5: This assumption requires $\mathcal{T}$ to be a contraction operator. This has been proven in theorem 1 of this paper. Theorem 3 in its current form is not general enough to provide convergence in all tabular settings. It is meant to provide a first way to prove convergence for recurrent learning and lead the way to more refined analysis. Such extensions are discussed later in the paper.

6.3.2 State dependent β

As mentioned earlier, state dependent β are responsible for the success of certain techniques. For instance, the success of LSTM Hochreiter and Schmidhuber 1997 can be attributed to the gating(state dependent smoothing) mechanism.In this section we consider β_ω where β_t is estimated with a set of parameters ω. There is a natural way to estimate β in our formulation as given below.

$$\min_\beta \mathop{\mathbb{E}}_{\pi} \left\| V^\pi - V_\theta^\beta \right\|_2 \tag{6.24}$$

In practice, this compares the estimate $V_\theta(s_t)$ and $V_{\theta,\omega}^\beta(s_{t-1})$ to V^π and gives more weight to the one that is closer among the two. This differs from the greedy approachM. White and A. White 2016 to set λ as we don't explicitly need to consider the variance of the return. As an example, we derive an analytic form of gradient of $\beta_\omega = \sigma(\omega_{s_t})$ where ω_{s_t} is a scalar. Taking the derivative of Eq. 6.24, gives the following update rule:

$$\begin{aligned} \omega_{s_t} = \omega_{s_t} + \alpha(\sigma(\omega_{s_t})(1 - \sigma(\omega_{s_t}) \\ (V^\pi(s_t) - V_\theta^\beta(s_t))(V_\theta(s_t) - V_\theta^\beta(s_{t-1}))). \end{aligned} \tag{6.25}$$

A full derivation of Eq. 6.25 is given below and a table illustrating the behavior of β following the loss described, can be found in section **??** of the appendix.

	$V^\pi(s_t) > V_\theta^\beta(s_t)$	$V^\pi(s_t) < V_\theta^\beta(s_t)$
$V(s_t) > V_\theta^\beta(s_{t-1})$	$\beta \uparrow$	$\beta \downarrow$
$V(s_t) < V_\theta^\beta(s_{t-1})$	$\beta \downarrow$	$\beta \uparrow$

We wish to find $\beta = \sigma(\omega)$ minimizing the loss :

$$\min \quad \frac{1}{2}(V^\pi(s_t) - V_{\theta,\omega}^\beta(s_t))^2 \tag{6.26}$$

$$\tag{6.27}$$

Taking the derivative of the R.H.S of 2 gives

$$\frac{d}{d\omega_{s_t}} \quad \left(\frac{1}{2}(V^\pi(s_t) - V_{\theta,\omega}^\beta(s_t))^2\right) = (V^\pi(s_t) - V_{\theta,\omega}^\beta(s_t))\Big(\frac{d(V^\pi(s_t) - V_{\theta,\omega}^\beta(s_t))}{d\omega_{s_t}}\Big)\Big) \quad \text{by chain rule} \tag{6.28}$$

We know that $\frac{d}{d\omega}\sigma(\omega_{s_t}) = \sigma(\omega_{s_t})(1 - \sigma(\omega_{s_t}))$
and $\frac{d}{d\sigma(\omega_{s_t})}(V^\pi(s_t) - V_{\theta,\omega}^\beta(s_t)) = \frac{d}{d\sigma(\omega_{s_t})}\left(\sigma(\omega_{s_t})V(s_t) + \Big(1-\sigma(\omega_{s_t})\Big)V_{\theta,\omega}^\beta(s_{t-1}) - V^\pi(s_t)\right) =$
$V(s_t) - V_{\theta,\omega}^\beta(s_{t-1})$

Therefore,

$$\frac{d}{d\omega} \quad \left(\frac{1}{2}(V^\pi(s_t) - V_{\theta,\omega}^\beta(s_t))^2 + \lambda\sigma(\omega_{s_t})\right) = \tag{6.29}$$

$$(V^\pi(s_t) - V_{\theta,\omega}^\beta(s_t))(V(s_t) - V_{\theta,\omega}^\beta(s_{t-1}))\left(\sigma(\omega_{s_t})(1 - \sigma(\omega_{s_t}))\right) + \lambda\left(\sigma(\omega_{s_t})(1 - \sigma(\omega_{s_t}))\right) = \tag{6.30}$$

$$\left(\sigma(\omega_{s_t})(1 - \sigma(\omega_{s_t}))\right)\left((V^\pi(s_t) - V_{\theta,\omega}^\beta(s_t))(V(s_t) - V_{\theta,\omega}^\beta(s_{t-1}))\right) \tag{6.31}$$

One caveat to consider with this update rule concerns the magnitude of V. In practice, to avoid the saturation of the sigmoid unit, the loss needs to be in a similar range $\sim [0, 1]$. This can be mitigated by scaling the loss or cutting the gradients above

a certain magnitude as done in PPO. Schulman, Wolski, et al. 2017. A second caveat is that the gradient with respect to ω could be expanded further to include previous time steps. This is ignored in the tabular setting to favor simplicity of the update. However, this gradient is considered in deep reinforcement learning setting due to the strength of automatic differentiation. We now give an example of how this error can behave. Consider a situation in which one of the two scenarios could happen. In the first scenario, we see a spike due to a noisy reward or an error in the function approximator or a noise in the state representation. In the second scenario, this spike arise due to a sparse reward from the environment. This is illustrated in the figure 6.1. In the first scenario, V_θ^β is a better estimate of the expected discounted return than a 1-step estimate. Therefore, the algorithm learns to lower β around the spike to smooth out the trajectory hence lowering the variance. However, in the second scenario, V_θ^β becomes biased and the algorithm learns to trust its current estimate, driving β to 1. This is a classic bias variance trade-off, where the state dependant β can help us lower the bias induced. There exists an interesting connection to explore between λ and β as λ return is a better target to learn β. This relationship is explored further in the experimental section of the paper.

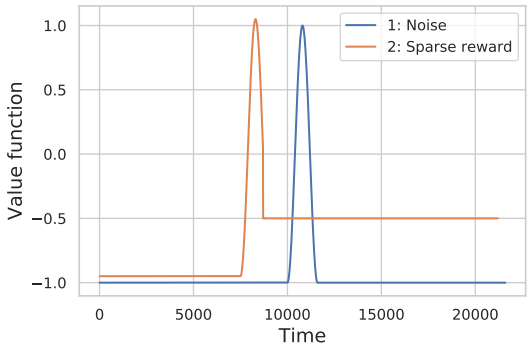


Figure 6.1: Noise versus sparse reward

6.3.3 Adjusting for the reward

In practice many environments in reinforcement learning have either a constant negative reward or a constant positive reward at every time step. In order to account for those rewards we propose an alternative formulation where V_θ^β accounts for the reward that was just seen:

$$V_\theta^\beta(s_t) = \beta_t V_\theta(s_t) + (1 - \beta_t)(V_\theta^\beta(s_{t-1}) - r_{t-1}) \tag{6.32}$$

The decision on the choice of formulation to use will depend on the environment considered. In the case where β is restricted to $\beta = \{0, 1\}$, the updates on a state $V(s_i)$ can be interpreted as an undiscounted N-step return:

$$\delta_t = V^\pi(s_t) + \sum_{p=i}^{t-1} r_p - V_\theta(s_i). \tag{6.33}$$

From a theoretical perspective, convergence of this formulation can be considered by modifying the bound on Δ to accommodate the reward. This is left as a future work.

6.4 Related work

Recurrent learning shares similarities with many algorithms in reinforcement learning. The most important similarity is with respect to λ return R. S. Sutton and Barto 1998; Dayan 1992. There is often a debate of explicit versus implicit modelling in reinforcement learning and supervised learning from a conceptual perspective. λ return is a way to implicitly enforce temporal coherence through the trajectory. In this paper, we propose a method to *explicitly* enforce the temporal coherence using recurrent learning. Furthermore, recurrent learning yields a natural way to estimate an emphasis function whereas setting λ efficiently still remains an open problem. In practice both λ return and recurrent learning may be needed to properly enforce temporal coherence. The capacity to ignore states also share some similar motivations to semi-Markov decision process Puterman 1990 and Temporal Value Transport Hung

et al. 2018. Temporal Value Transport attempts to exploit similar ideas that recurrent learning does. It does so in a different manner. In particular, Temporal Value Transport is based on a discrete attention mechanism using threshold values in contrast to our continuous β *attention* mechanism. This yields very different algorithms and theory in practice. As mentioned earlier, there exists similarities with emphatic TD Mahmood et al. 2015 in the sense that it emphasizes or de-emphasizes the update done to a state based on β. One key difference with emphatic TD is that the interest is decayed across all the states using γ, whereas in this work it is based on the interest of the future states. Our formulation is driven by the success of *forgetting* and *ignoring* in supervised learning. Furthermore, learning the emphasis function is not explored in Mahmood et al. 2015. Our work also relates to Temporal Regularization Thodoroff et al. 2018 that smooth the target of temporal difference methods using previous values of the trajectory. Although sharing similar motivation, the algorithms proposed are different. In particular, learning *smoothing* coefficient is not considered. Finally, Xu et al. 2017 proposes a mechanism to adaptively learn to use previous estimates. In practice, this is done by considering an *auxiliary loss* between the target and the previous estimate in contrast with the setup considered here. Furthermore, the theoretical properties(convergence) of their algorithm is not considered.

6.5 Experiments

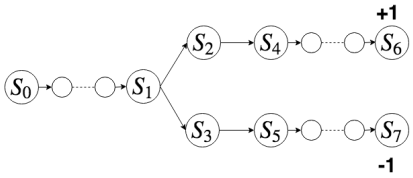


Figure 6.2: Simple chain MDP. The dotted line in the figure indicates multiple states in between shown states.

We consider the simple chain MDP described in figure 6.2 to demonstrate our

method. This MDP has three chains connected together to form a Y. Each of the 3 chains (left of S_1, right of S_2, right of S_3) is made up of a sequence of states. The agent starts at S_0 and navigates through the chain. At the intersection, S_1, there is a 0.5 probability to go top or bottom. The chain on the top has a reward of $+1$ at the last state and the chain on the bottom has a reward of -1. Every other transition has a reward of 0, unless specified otherwise.

6.5.1 Bias Variance trade-off

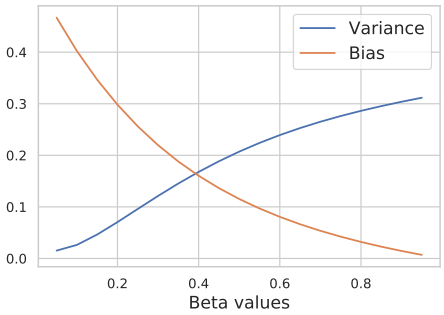


Figure 6.3: Bias variance trade off of V_θ^β throughout the trajectory using exponential smoothing

It is possible to decompose policy evaluation into two sub tasks. The first one is given a set of functions V_θ, the goal is to estimate optimal values of states along the trajectory. The second task is to compare this set of functions that can be estimated by either considering the loss in Eq. 6.2 or the recurrent loss in Eq. 6.8. In this experiment, we focus on the former task to demonstrate the bias variance trade-off induced by exponentially smoothing the values over the trajectory. This is shown in figure 6.3. The value function V_θ is learned using TD(0) with the loss described in Eq. 6.2. For all states $s_i \in \mathbb{S}$ we average the bias and the variance of $V_\theta^\beta(s_i)$ over 500 episodes. We define the *bias* as $\left\|V_\theta(s_i) - V_\theta^\beta(s_i)\right\|_2$. Intuitively, it is the distance between the estimate $V_\theta(s_i)$(used as optimal values for this experiment) and the exponentially smoothed estimate $V_\theta^\beta(s_i)$. We experiment with a range of fixed

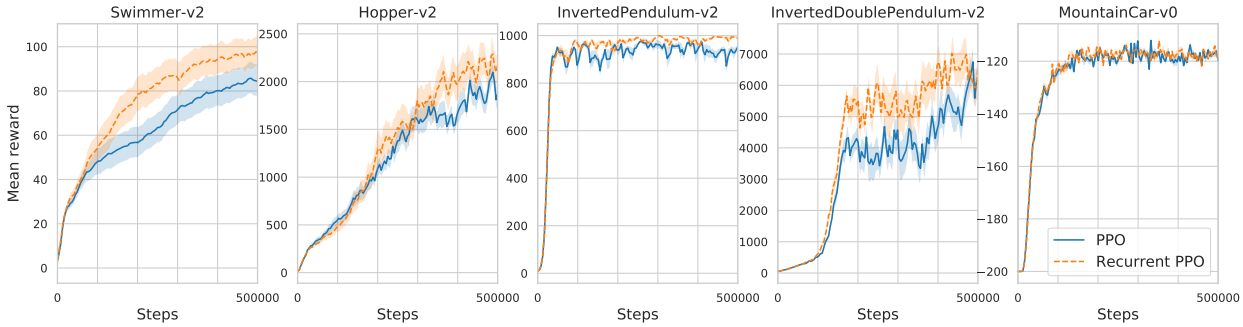


Figure 6.4: Performance on Mujoco environment. The X-axis represents the training steps and the Y-axis represents the mean reward

β's. We observe a decrease in bias and an increase in the variance as the β increases. In other words, the value of each state is forced to be close to the values in the past when β is low. This results in smooth values along the trajectory but biased due to their reliance on the past. As $\beta \to 1$ we tend to be close to classic TD setting. One important note, to accurately demonstrate the bias variance trade-off induced by using exponential smoothing, we use the same learning mechanism for both V_θ^β and V_θ, namely the loss in Eq. (6.2). This choice is necessary to have a proper comparison; choosing different losses would induce different fixed points and make the comparison difficult.

6.5.2 Partially observable setting

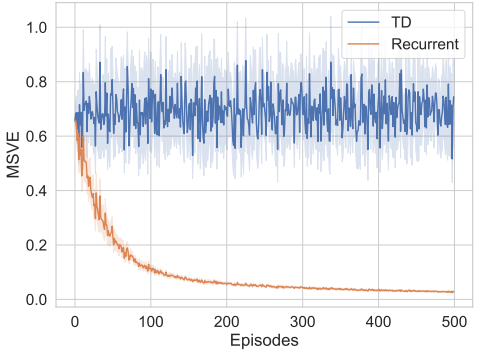
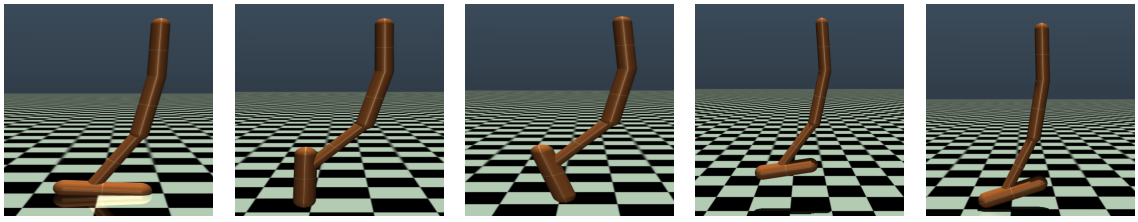


Figure 6.5: TD versus Recurrent TD on aliased Y-chain

Next, we explore the capacity of recurrent learning to solve a partially observable task. In particular, we consider the case where some states are aliased (share a common representation). The representation of the state following S_4 and S_5 in figure 6.2 is aliased. The goal of this environment is to correctly estimate the value of the aliased state $V^\pi(S_4) = 0.81, V^\pi(S_5) = -0.81$(due to the discounting and the length of each chain being 5). When TD(λ) is used to estimate the values for these states, the values of states S_4 and S_5 is close to 0 as the reward at the end of the chain is $+1$ and -1. However, when learning β, recurrent learning achieve almost no error in the estimate of the aliased state as illustrated in figure 6.5. To understand this result the first thing to realize is that $\beta = 0$ on the aliased state as the previous values along the trajectory are a better estimate of the future along the same chain compared to the actual estimate of the aliased state. As $\beta \to 0$, $V_\theta^\beta(S_4)$ and $V_\theta^\beta(S_5)$ tends to rely on the previous estimate $V(S_2)$ and $V(S_3)$ which are accurate. We see that learning to ignore certain states can sometimes be enough to solve correctly an aliased tasks in contrast to a traditional POMDP method that would attempt to infer the belief state. The results displayed in figure 6.5 are averaged over 20 random seeds. The learning rate used is 0.05, $\gamma = 0.9$ and $\lambda = 0.9$.

6.5.3 Deep reinforcement learning



Phase 1: high β | Phase 2: low β | Phase 3: high β | Phase 4: low β | Phase 1: high β

Figure 6.6: Visual illustration of the cyclical behavior of β on Hopper

In this experiment, we modify the critic of Proximal Policy Optimization (PPO) Schulman, Wolski, et al. 2017 to use recurrent learning. We modify the critic to estimate

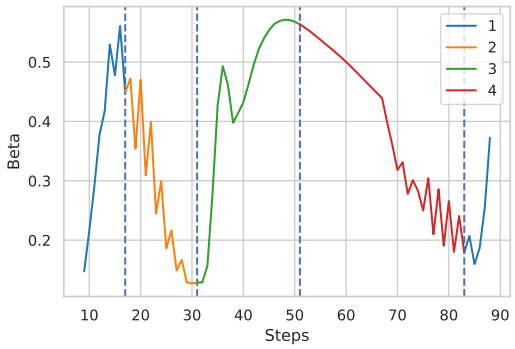


Figure 6.7: Behavior of β through the trajectory on Hopper with the different phase described in figure 6-10.

the value function parametrized by θ using recurrent learning. We add a separate network parametrized by ω (same architecture as PPO) to learn a state dependant β. The following loss is minimized:

$$\min_{\beta,\omega} \mathbb{E}_{\pi}(V_\theta^\lambda - V_\theta^\beta)^2) \tag{6.34}$$

where the target V_θ^λ is the generalized advantage function Schulman, Moritz, et al. 2015. Using an automatic differentiation library (Pytorch Paszke et al. 2017) we differentiate Eq.6.34 through the modified estimate to learn the θ and ω. The hyper-parameters of PPO are not modified. Due to the batch updates in PPO, obtaining the trajectory information to create the computational graph can be costly. As a result, we cut the backpropagation after N timesteps in a similar manner to truncated backpropagation through time Williams and Zipser 1995. The learning rate for the β network, L1 regularization coefficient of β and number of backpropagation steps were obtained using hyperparameter search and details can be found in the appendix **??**. We used a truncated backprop of $N = 5$ in our experiments as found no empirical improvements for $N = 10$. However, we expect in some settings(e.g. long term credit assignment) backprop further can be advantageous. The motivation to regularize β to be sparse comes from bounded rationality. Our brain has limited capacity and only updating the value at key states is a desirable property. This parameter plays a similar role as the deliberation cost in option Harb et al. 2018. The best performing set

of parameters included a regularization parameter of 0.5 supporting this hypothesis. The performance reported are averaged over 20 different random seeds. [1].

6.5.3.1 Performance

The algorithm was tested on seven different games of the Mujoco suite Todorov, Erez, and Tassa 2012. As demonstrated in the figure 6.4, we observe an increase in performance on four of the tasks (Swimmer, Hopper, Inverted Pendulum, Double Inverted Pendulum). The performances are averaged over 20 seeds and a confidence interval of 95% is used. For the other three tasks (Half Cheetah, Walker, Mountain Car) the performances were found to be comparable. The performance graphs for all the games can be found in figure **??** in the appendix. The mean beta values and variance for all the tasks can also be found in the appendix (figure **??**,**??**). One interesting finding in the figure 6.4 concerns inverted-pendulum in the context of generalization. Though the convergence is quick for PPO, severe performance drops (reward drop by more than 100) can be observed. We found that recurrent learning stabilizes the learning and suffer significantly less drops during training. We notice that on an average PPO will suffer from 5.91 drops below 900 points over 500k steps. In contrast, recurrent learning will only drop 3.53 times. This represent a 40% drop in *catastrophic forgetting* underlying the potential robustness of recurrent learning.

6.5.3.2 Qualitative interpretation of β

Hopper: At the end of training, we qualitatively analyze β through the trajectory and observe the cyclical behavior shown in figure 6.7, where different colors describe various stages of the cycle. One intuitive way to look at β is: *if I were to give a different value to a state would that alter my policy significantly?* We observe an increase in β value when the agent has to take an important decision like jumping or

[1]The base code used to develop this algorithm can be found here Kostrikov 2018. The code for Recurrent PPO can be found in the supplementary material

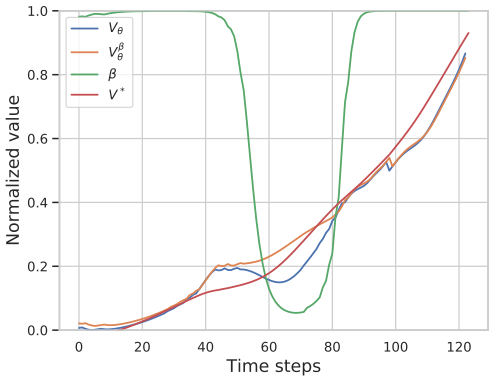


Figure 6.8: Behavior of β and the value function on Mountain-Car

landing. We see a decrease in β when the agent has trivial actions to perform. This pattern is illustrated in the figure 6.6 and 6.7. This behaviour is cyclic and repetitive and a video of the same can be found at the following link[2]. One surprising fact was that this behavior was obtained without any regularization on β. Similar behavior can be observed with regularization although the mean and variance of β diminishes.

Mountain car: Two scenarios may happen when the agent is climbing up the hill on the right side. Either the agent has enough velocity to finish the game and obtain a high reward, or it doesn't have enough velocity and goes back down the hill. During early stages of training, the function approximator is confused about the scenarios mentioned earlier, resulting a drop in value function around step 100 as shown in figure 6.8. The value increases again once the agent climbs the hill with more velocity. In PPO, we can obtain a more accurate target by setting τ to a high value, thereby eliminating a drop in value. This enables the β network to learn to trust its past estimate rather than the noisy point estimate, hence a significant drop in the β value. As a result, V_θ^β becomes a better estimate of the target than V_θ in this scenario. After training PPO for a while this drop disappears and the β mean goes to 1. This experiment shows the potential of β to smooth out noisy estimates in the trajectory. One caveat to consider is the feedback loop induced by ignoring a state in control. When the policy changes a state that can be ignored at the beginning may be essential

[2] https://youtu.be/0bzEcrxNwRw

later on. One way to address this is to avoid saturating β such that learning remains possible later on.

6.6 Discussions

6.6.1 β as an interest function:

One interesting result of this work is that the β network learns to ignore some state without any restrictions imposed on it. It does so in order to reduce the variance. Furthermore, this β can be interpreted as an *interest* function. In reinforcement learning, having access to a function quantifying the *interest* Mahmood et al. 2015 of a state can be helpful. For example, one could decide to explore from those states, prioritize experience replay based on those states. Further work can be done to study how β may impact performance. We also believe β can be related to the concepts of bottleneck state Tishby and Polani 2011. Finally, the concept of interest state in recurrent learning aligns well with the notion of interest state for the λ return. Indeed bootstrapping on states with similar values than the one estimated will only result in variance. The most informative updates comes from bootstrapping on state with different value.

6.6.2 Partial observability:

As demonstrated in the experiments earlier, recurrent learning is able in some cases to correctly estimate the value of aliased state using trajectory's estimate. This is a promising area to research as the experiments suggest that ignoring an uninformative state can sometimes be enough to learn its value function. This is in contrast to traditional POMDP method that attempts to infer the belief state. Inferring the true underlying state is not always feasible. In some cases learning to ignore aliased state and relying on previous estimates may be an easier task.

6.6.3 Relationship between λ and β:

There exists an important relationship between λ and β. β can choose to ignore a state if the past is different from the future. We could get a better estimate of the future using λ returns when compared against one step methods. Hence, a need for long horizon to learn accurate β. This behaviour can also be seen in our *mountain car* experiment. Also, bootstrapping from states that are ignored would be detrimental to the learning process. A state is never updated if it has $\beta = 0$ hence bootstrapping from this state will induce *wrong* bias. To avoid this issue, one approach could be to use learned β as a state dependent λ hence making correct updates.

6.6.4 Asymptotic convergence:

As mentioned previously assumption 2 is too restrictive to guarantee convergence in all tabular settings. This analysis could be refined by considering the trajectory's information to bound Δ. As an example, one could consider any physical system where transition in the state space are smooth (continuous state space) and bounded by some Lipschitz constant in a similar manner than Shah and Xie 2018.

6.6.5 Bias of the eligibility trace:

We showed in Eq. (6.11) a technique to compute the gradient of the estimate recursively. This technique is computationally inexpensive compared to the recursive backpropogation as the gradient from the past is accumulated. But this kind of an update is biased. This bias comes from the fact that the true gradient obtained by applying the chain rule at future step is different than the trace if the parameters are updated at each time step. This is a similar problem encountered by eligibility traces Seijen and R. Sutton 2014 and real time recurrent learning Williams and Zipser 1995. Similar idea than Seijen and R. Sutton 2014 may be used to obtain an unbiased estimate.

6.6.6 Recurrent learning for action:

We have not explored the concept of recurrent learning for actions (Q values and policy gradient). This is a promising area as constraining actions to be temporally coherent is a natural prior to induce on a function approximator. This technique could also be interesting in the context of exploration as this could yield a *structured exploration*. Finally, this framework with learning β can be cast as a *vanilla* version of options R. S. Sutton, Precup, and S. Singh 1999.

Conclusion: This paper proposes a technique to address two important aspects of reinforcement learning algorithms namely - temporal coherence, and selective updating. First, we prove asymptotic convergence of the proposed method. Later, we provide a bias-variance argument to emphasize the importance of temporal coherence. Then, we demonstrate interesting properties that result while we emphasize and de-emphasize updates on states during learning. We provide experiments to corroborate the application of our method in a partially observable domain and for continuous control.

7

Conclusion

This paper tackles the problem of regularization in RL from a new angle, that is from a temporal perspective. In contrast with typical spatial regularization, where one assumes that rewards are close for nearby states in the state space, temporal regularization rather assumes that rewards are close for states *visited closely in time*. This approach allows information to propagate faster into states that are hard to reach, which could prove useful for exploration. The robustness of the proposed approach to noisy state representations and its interesting properties should motivate further work to explore novel ways of exploiting temporal information.

Bibliography

Abbeel, Pieter, Adam Coates, and Andrew Y Ng (2010). "Autonomous helicopter aerobatics through apprenticeship learning". In: *The International Journal of Robotics Research* 29.13, pp. 1608–1639.

Banach, Stefan (1922). "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales". In:

Bartlett, Peter L and Ambuj Tewari (2009). "REGAL: A regularization based algorithm for reinforcement learning in weakly communicating MDPs". In: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, pp. 35–42.

Baxter, Jonathan and Peter L Bartlett (2001). "Infinite-horizon policy-gradient estimation". In: *Journal of Artificial Intelligence Research* 15, pp. 319–350.

Bellemare, Marc G et al. (2013). "The arcade learning environment: An evaluation platform for general agents". In: *Journal of Artificial Intelligence Research* 47, pp. 253–279.

Borkar, Vivek S (2009). *Stochastic approximation: a dynamical systems viewpoint*. Vol. 48. Springer.

Borkar, Vivek S and Sean P Meyn (2000). "The ODE method for convergence of stochastic approximation and reinforcement learning". In: *SIAM Journal on Control and Optimization* 38.2, pp. 447–469.

Box, George, Gwilym M. Jenkins, and Gregory C. Reinsel (1994). *Time Series Analysis: Forecasting and Control (3rd ed.)* Prentice-Hall.

Brémaud, Pierre (2013). *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*. Vol. 31. Springer Science & Business Media.

Chung, Junyoung et al. (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555*.

Chung, Kai-Min et al. (2012). "Chernoff-Hoeffding bounds for Markov chains: Generalized and simplified". In: *arXiv preprint arXiv:1201.0559*.

Dayan, Peter (1992). "The convergence of TD (λ) for general λ". In: *Machine learning* 8.3-4, pp. 341–362.

Dhariwal, Prafulla et al. (2017). *OpenAI Baselines*. https://github.com/openai/baselines.

Dhingra, Bhuwan et al. (2017). "Towards End-to-End Reinforcement Learning of Dialogue Agents for Information Access". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Vol. 1, pp. 484–495.

Diaconis, Persi and Daniel Stroock (1991). "Geometric bounds for eigenvalues of Markov chains". In: *The Annals of Applied Probability*, pp. 36–61.

Farahmand, Amir-massoud (2011). "Regularization in reinforcement learning". PhD thesis. University of Alberta.

Farahmand, Amir-massoud et al. (2009). "Regularized fitted Q-iteration for planning in continuous-space Markovian decision problems". In: *American Control Conference*. IEEE, pp. 725–730.

Fill, James Allen et al. (1991). "Eigenvalue Bounds on Convergence to Stationarity for Nonreversible Markov Chains, with an Application to the Exclusion Process". In: *The Annals of Applied Probability* 1.1, pp. 62–87.

François-Lavet, Vincent et al. (2016). "Deep reinforcement learning solutions for energy microgrids management". In: *European Workshop on Reinforcement Learning (EWRL 2016)*.

Gardner Jr, Everette S (1985). "Exponential smoothing: The state of the art". In: *Journal of forecasting* 4.1, pp. 1–28.

Gardner, Everette S (2006). "Exponential smoothing: The state of the art—Part II". In: *International journal of forecasting* 22.4, pp. 637–666.

Golub, Gene H and Richard S Varga (1961). "Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods". In: *Numerische Mathematik* 3.1, pp. 147–156.

Grinberg, Yuri, Doina Precup, and Michel Gendreau (2014). "Optimizing energy production using policy search and predictive state representations". In: *Advances in Neural Information Processing Systems*, pp. 2969–2977.

Harb, Jean et al. (2017). "When waiting is not an option: Learning options with a deliberation cost". In: *arXiv preprint arXiv:1709.04571*.

— (2018). "When waiting is not an option: Learning options with a deliberation cost". In: *Thirty-Second AAAI Conference on Artificial Intelligence*.

Harrigan, Cosmo (2016). "Deep reinforcement learning with regularized convolutional neural fitted q iteration". In:

Henderson, Peter et al. (2017). "Deep reinforcement learning that matters". In: *arXiv preprint arXiv:1709.06560*.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hung, Chia-Chun et al. (2018). *Optimizing Agent Behavior over Long Time Scales by Transporting Value*.

Kearns, Michael J and Satinder P Singh (2000). "Bias-Variance Error Bounds for Temporal Difference Updates." In:

Kemeny, John G and James Laurie Snell (1976). "Finite markov chains, undergraduate texts in mathematics". In:

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kober, Jens, J Andrew Bagnell, and Jan Peters (2013). "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* 32.11, pp. 1238–1274.

Koedinger, K.R. et al. (2018). "New Potentials for Data-Driven Intelligent Tutoring System Development and Optimization". In: *AAAI magazine*.

Konda, Vijay R and John N Tsitsiklis (2000). "Actor-critic algorithms". In: *Advances in neural information processing systems*, pp. 1008–1014.

Kostrikov, Ilya (2018). *PyTorch Implementations of Reinforcement Learning Algorithms*. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr.

Leike, Jan (2016). "Nonparametric General Reinforcement Learning". In: *arXiv preprint arXiv:1611.08944*.

Levin, David A and Yuval Peres (2008). *Markov chains and mixing times*. Vol. 107. American Mathematical Soc.

Liu, Bo, Sridhar Mahadevan, and Ji Liu (2012). "Regularized off-policy TD-learning". In: *Advances in Neural Information Processing Systems*, pp. 836–844.

Mahmood, A Rupam et al. (2015). "Emphatic temporal-difference learning". In: *arXiv preprint arXiv:1507.01569*.

Mnih, Volodymyr, Adria Puigdomenech Badia, et al. (2016). "Asynchronous methods for deep reinforcement learning". In: *International Conference on Machine Learning*, pp. 1928–1937.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, et al. (2013). "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602*.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, p. 529.

Neu, Gergely, Anders Jonsson, and Vicenç Gómez (2017). "A unified view of entropy-regularized markov decision processes". In: *arXiv preprint arXiv:1705.07798*.

Norris, James R (1998). *Markov chains*. 2. Cambridge university press.

Paszke, Adam et al. (2017). "Automatic differentiation in PyTorch". In: *NIPS-W*.

Pazis, Jason and Ronald Parr (2011). "Non-Parametric Approximate Linear Programming for MDPs." In: *AAAI*.

Petrik, Marek et al. (2010). "Feature selection using regularization in approximate linear programs for Markov decision processes". In: *arXiv preprint arXiv:1005.1860*.

Polyak, Boris T and Anatoli B Juditsky (1992). "Acceleration of stochastic approximation by averaging". In: *SIAM Journal on Control and Optimization* 30.4, pp. 838–855.

Prasad, N. et al. (2017). "A Reinforcement Learning Approach to Weaning of Mechanical Ventilation in Intensive Care Units". In: *UAI*.

Precup, Doina (2000). "Eligibility traces for off-policy policy evaluation". In: *Computer Science Department Faculty Publication Series*, p. 80.

Puterman, Martin L (1990). "Markov decision processes". In: *Handbooks in operations research and management science* 2, pp. 331–434.

— (1994). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley Sons.

Schulman, John, Sergey Levine, et al. (2015). "Trust region policy optimization". In: *International Conference on Machine Learning*, pp. 1889–1897.

Schulman, John, Philipp Moritz, et al. (2015). "High-dimensional continuous control using generalized advantage estimation". In: *arXiv preprint arXiv:1506.02438*.

Schulman, John, Filip Wolski, et al. (2017). "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347*.

Seijen, Harm and Rich Sutton (2014). "True online TD (lambda)". In: *International Conference on Machine Learning*, pp. 692–700.

Shah, Devavrat and Qiaomin Xie (2018). "Q-learning with Nearest Neighbors". In: *arXiv preprint arXiv:1802.03900*.

Shortreed, S. et al. (2011). "Informing sequential clinical decision-making through reinforcement learning: an empirical study". In: *Machine Learning*.

Silver, D. et al. (2016). "Mastering the game of Go with deep neural networks and tree search". In: *Nature*.

Singh, Satinder P and Richard S Sutton (1996). "Reinforcement learning with replacing eligibility traces". In: *Machine learning* 22.1-3, pp. 123–158.

Sutton, Richard S (1988). "Learning to predict by the methods of temporal differences". In: *Machine learning* 3.1, pp. 9–44.

Sutton, Richard S and Andrew G Barto (n.d.). "Reinforcement Learning: An Introduction". In: ().

— (1998). *Reinforcement learning: An introduction*. 1st. MIT press Cambridge.

— (2017). *Reinforcement learning: An introduction*. (in progress) 2nd. MIT press Cambridge.

Sutton, Richard S, Anna Koop, and David Silver (2007). "On the role of tracking in stationary environments". In: *Proceedings of the 24th international conference on Machine learning*. ACM, pp. 871–878.

Sutton, Richard S, A Rupam Mahmood, and Martha White (2016). "An emphatic approach to the problem of off-policy temporal-difference learning". In: *The Journal of Machine Learning Research* 17.1, pp. 2603–2631.

Sutton, Richard S, David A McAllester, et al. (2000). "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in neural information processing systems*, pp. 1057–1063.

Sutton, Richard S, Doina Precup, and Satinder Singh (1999). "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". In: *Artificial intelligence* 112.1-2, pp. 181–211.

Sutton, Richard S and Satinder P Singh (n.d.). "On step-size and bias in temporal-difference learning". In: Citeseer.

Sutton, Richard Stuart (1984). "Temporal credit assignment in reinforcement learning". In:

Thodoroff, Pierre et al. (2018). "Temporal Regularization for Markov Decision Process". In: *Advances in Neural Information Processing Systems*, pp. 1782–1792.

Tishby, Naftali and Daniel Polani (2011). "Information theory of decisions and actions". In: pp. 601–636.

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). "Mujoco: A physics engine for model-based control". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pp. 5026–5033.

Tsitsiklis, John N (1994). "Asynchronous stochastic approximation and Q-learning". In: *Machine learning* 16.3, pp. 185–202.

Tsitsiklis, John N and Benjamin Van Roy (2002). "On average versus discounted reward temporal-difference learning". In: *Machine Learning* 49.2-3, pp. 179–191.

Varga, Richard S (2009). *Matrix iterative analysis*. Vol. 27. Springer Science & Business Media.

Vinyals, Oriol et al. (2017). "Starcraft ii: A new challenge for reinforcement learning". In: *arXiv preprint arXiv:1708.04782*.

Watkins, Christopher JCH and Peter Dayan (1992). "Q-learning". In: *Machine learning* 8.3-4, pp. 279–292.

White, Martha and Adam White (2016). "A greedy approach to adapting the trace parameter for temporal difference learning". In: *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, pp. 557–565.

Williams, Ronald J and David Zipser (1995). "Gradient-based learning algorithms for recurrent networks and their computational complexity". In: *Backpropagation: Theory, architectures, and applications* 1, pp. 433–486.

Wu, Yuhuai et al. (2017). "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation". In: *Advances in neural information processing systems*, pp. 5279–5288.

Xu, Zhongwen et al. (2017). "Natural Value Approximators: Learning when to Trust Past Estimates". In: *Advances in Neural Information Processing Systems*, pp. 2117–2125.