Exploiting trajectory information in Reinforcement Learning

Pierre Thodoroff

Computer Science
McGill University, Montreal

May 30, 2019

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Master of Science. ©Pierre Thodoroff; May 30, 2019.

Acknowledgements

TODO

Abstract

Model-free Reinforcement Learning (RL) is a widely used framework for sequential decision making in many domains such as robotics and video games. However, its use in the real-world remains limited due, in part, to the high variance of value function estimates, leading to poor sample complexity.

The problem of disentangling signal from noise in sequential domains is not specific to Reinforcement Learning and has been extensively studied in the Supervised Learning literature. In this thesis we propose to view the value function through the trajectory as a Time-Series. We leverage the Time-Series literature to propose methods aimed at reducing the variance of value function estimates.

Résumé

L'apprentissage par renforcement est un model d'apprentissage fréquemment utilise pour modelise les problems de decisions sequentielles comme la robotique et les jeux videos. Cependent son utilisation dans le monde reele reste limite a cause de la forte variance des estimation de valeur, resultant

Contents

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.

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. In this thesis we focus on value based reinforcement learning. INTRO VALUE BASED .. 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. We propose 2 method to mitigate those issue 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

1.2 Introduction

Reinforcement learning algorithms often suffers from variance 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.

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 $P_{ij} = \mathcal{P}(X_{n+1} = j | X_n =$

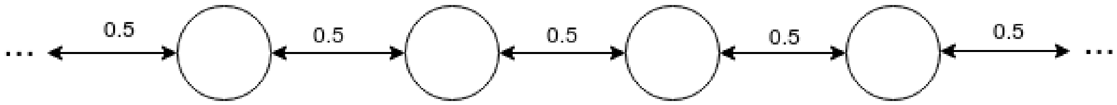


Figure 2.1: Random walk

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. 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:*

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

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 challenges 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 $\mu \quad \forall\beta \in [0,1]$.*

Proof.

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

□

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 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

```
Input: $\pi, \gamma$
for all steps do
    Choose $a \sim \pi(\mathcal{S})$
    Take action $a$, observe $r(s), s'$
    $V(s) = r(s) + \gamma V(s')$
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^π_t 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

```
Input: $\pi, \alpha, \gamma$
for all steps do
    Choose $a \sim \pi(\mathcal{S})$
    Take action $a$, observe $r(s), s'$
    $V(s) = V(s) + \alpha(r(s) + \gamma V(s') - V(s))$
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 Singh, 2000; R. S. Sutton and 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; Singh and R. S. Sutton, 1996; Precup, 2000).

Algorithm 3 Temporal Difference with eligibility traces

```
Input: $\pi, \alpha, \gamma, \lambda$
for all steps do
    Choose $a \sim \pi(\mathcal{S})$
    Take action $a$, observe $r(s), s'$
    $e(t) = \nVdash_s + \gamma\lambda e(t-1)$
    $V = V + \alpha(r(s) + \gamma V(s') - V(s))e(t)$
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. 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:

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{\pi}[(V^\pi - V_\theta)^2] \tag{3.13}$$

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

$$\theta_{t+1} = \theta_t + \alpha(V^\pi(s_t) - V_{\theta_t}(s_t))\nabla_{\theta_t} V_\theta(s_t) \tag{3.14}$$

where α is a learning rate.

In practice V^π is approximated using Monte Carlo rollouts (R. S. Sutton and Barto, n.d.) or TD methods (R. S. Sutton, 1988).

3.2.5 Convergence using Stochastic Approximation

Convergence in the stochastic setting is usually proven by casting the learning algorithm as a stochastic approximation (Tsitsiklis, 1994; Borkar, 2009; Borkar and Meyn, 2000) of the form:

$$\theta_{t+1} = \theta_t + \alpha(\mathcal{T}\theta_t - \theta_t + w(t)) \tag{3.15}$$

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):

$$\mathcal{T}V_\theta(s_t) = \mathop{\mathbb{E}}_{s_{t+1}\sim\pi}[r(s_t) + \gamma V_\theta(s_{t+1})] \tag{3.16}$$

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

$$w(t) = r_t + \gamma V_\theta(s_{t+1}) - \mathop{\mathbb{E}}_{s_{t+1}\sim\pi}[r + \gamma V_\theta(s_{t+1})] \tag{3.17}$$

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:

$$\mathcal{T}^* v^* = \max_{a\in\mathcal{A}}[r(a) + \gamma P(a)v^*]. \tag{3.18}$$

3.3.2 Policy Iteration

Using the operator defined in eq 3.18, 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: $\pi, \alpha, \gamma$
Policy Evaluation:
    for all steps do
        Choose $a \sim \pi(\mathcal{S})$
        Take action $a$, observe $r(s), s'$
        $V(s) = r(s) + \gamma V(s')$
    end for
Policy Improvement:
    for all $s \in \mathcal{S}$ do
        $\pi(s) \leftarrow \text{argmax}_a \sum_{s'} p(s', r|s, a) + [r + \gamma V(s')]$
    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.

$$\begin{aligned} Q(s,a) &= \mathop{\mathbb{E}}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s = s_0, a = a_0] \\ &= \mathop{\mathbb{E}}_{s'}[r_t + \gamma v(s')] \\ &= \mathop{\mathbb{E}}_{s'}[r_t + \gamma \text{max}_{a'} Q(s', a')] \end{aligned} \tag{3.19}$$

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

```
Input: $\pi, \alpha, \gamma$
for all steps do
    Choose $a \sim \pi(\mathcal{S})$
    Take action $a$, observe $r(s), s'$
    $Q(s,a) = Q(s,a) + \alpha(r(s) + \gamma Q(s', a') - Q(s,a))$
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:*

$$\nabla J(\theta) = \int_s d^\pi(s) \int_a \nabla_\theta \pi(a,s) Q^\pi(s,a) \tag{3.20}$$

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,

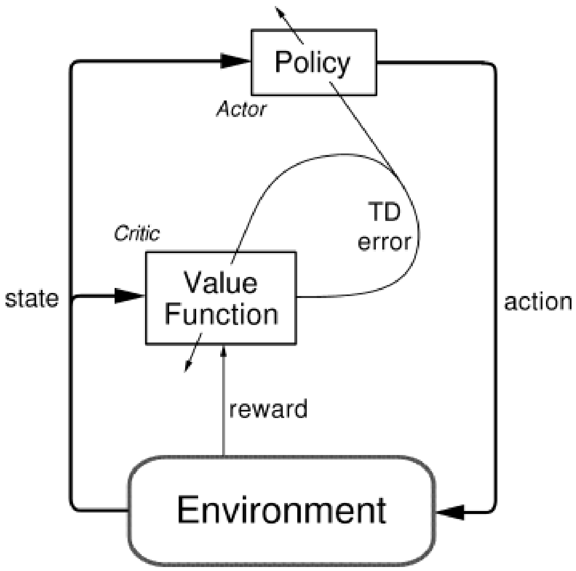


Figure 3.1: Actor-critic

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.

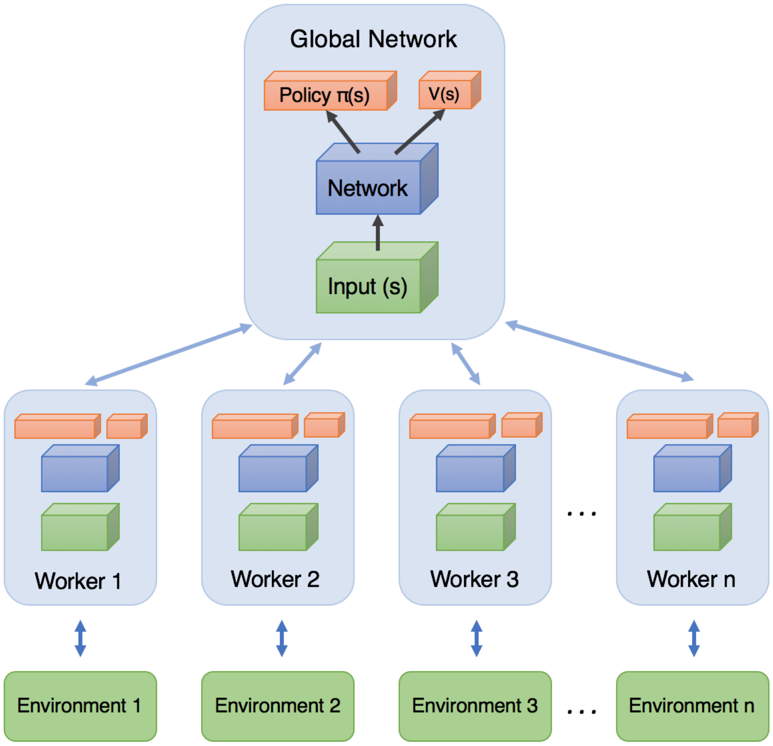


Figure 3.2: A3C diagram from LOOK AT COPYRIGHT

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:

$$\beta \mathrm{KL}(\pi_{\text{old}}, \pi_{\text{new}}) \tag{3.21}$$

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.

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

3.5 Regularization

Regularization in RL has been considered in several different perspectives. In this section we discuss the most popular one.

3.5.1 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. For example Farahmand, 2011 introduce Approximate Value iteration, designed to apply regularization technique to select value function estimators from rich function space.

3.5.2 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. 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.

$$\begin{aligned} \pi &= \mathop{\mathbb{E}}_{\pi} r + R(\pi) \\ R(\pi) &= \text{entrop} \end{aligned} \tag{3.22}$$

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.

3.5.3 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.

3.5.3.1 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.

3.5.3.2 Backward bootstrapping

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

3.5.4 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.

3.5.4.1 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.

4

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.

4.1 Value based temporal regularization

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:

$$\begin{aligned}V_\beta(s_t) &= \mathop{\mathbb{E}}_{s_{t+1},s_{t-1}\sim\pi}[r(s_t) + \gamma((1-\beta)V_\beta(s_{t+1}) + \beta V_\beta(s_{t-1}))] \\ &= r(s_t) + \gamma(1-\beta)\sum_{s_{t+1}\in\mathcal{S}} p(s_{t+1}|s_t)V_\beta(s_{t+1}) + \gamma\beta \sum_{s_{t-1}\in\mathcal{S}} \frac{p(s_t|s_{t-1})p(s_{t-1})}{p(s_t)} V_\beta(s_{t-1}),\end{aligned} \tag{4.1}$$

for a parameter $\beta \in [0,1]$ and $p(s_{t+1}|s_t)$ the transition probability induced by the policy π. 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:

$$\mathcal{T}_\beta^\pi v_\beta = r + \gamma((1-\beta)P^\pi v_\beta + \beta\widetilde{P^\pi} v_\beta). \tag{4.2}$$

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

Remark. *For $\beta = 0$, Eq. 4.2 corresponds to the original Bellman operator.*

We can prove that this operator has the following property.

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

$$\begin{aligned}\left\|\mathcal{T}_\beta^\pi u - \mathcal{T}_\beta^\pi v\right\|_\infty &= \left\|r + \gamma((1-\beta)Pu + \beta\widetilde{P}u) - (r + \gamma((1-\beta)Pv + \beta\widetilde{P}v))\right\|_\infty \\ &= \gamma\left\|((1-\beta)P + \beta\widetilde{P})(u-v)\right\|_\infty \\ &\leq \gamma\left\|u-v\right\|_\infty,\end{aligned} \tag{4.3}$$

where the last inequality uses the fact that the convex combination of two row stochastic matrices is also row stochastic (the proof can be found in lemma 3). Then using Banach fixed point theorem, we obtain that V_β^π is a unique fixed point. □

Lemma 3. *The convex combination of two row stochastic matrices is also row stochastic.*

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

$$\begin{aligned}(\beta P^{\pi} + (1-\beta)\widetilde{P^{\pi}})e &= \beta P^{\pi} e + (1-\beta)\widetilde{P^{\pi}} e \\ &= \beta e + (1-\beta) e \\ &= e.\end{aligned} \tag{4.4}$$

□

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 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

$$\begin{aligned}\mu((1-\beta)P^{\pi} + \beta\widetilde{P^{\pi}}) &= (1-\beta)\mu P^{\pi} + \beta\mu\widetilde{P^{\pi}} \\ &= (1-\beta)\mu + \beta\mu \\ &= \mu.\end{aligned} \tag{4.5}$$

□

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.

proposition 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*

$$\left\|v^{\pi} - v^{\pi}_{\beta}\right\|_{\infty} = \left\|\sum_{i=0}^{\infty} \gamma^i (P^i - ((1-\beta)P + \beta\widetilde{P})^i) r\right\|_{\infty} \leq \sum_{i=0}^{\infty} \gamma^i \left\|(P^i - ((1-\beta)P + \beta\widetilde{P})^i) r\right\|_{\infty}. \tag{4.6}$$

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 4, 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* $v^\pi = v^\pi_\beta$*.*

4.1.1 Discounted average reward case:

The temporally regularized MDP has the same discounted average reward as the original one as it is possible to define the discounted 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).

proposition 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 reweighs the reward on each state based on the Markov reversal, while preserving the average reward.

4.1.2 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 $(\widetilde{v_0}, \widetilde{v_1}...\widetilde{v_{n-1}})$:

$$V(s_t) = r(s) + \gamma \sum_{i=0}^{n-1} [\beta(i)\widetilde{V}_i(s_{t+1})], \tag{4.7}$$

where $\widetilde{V}_0(s_{t+1}) = V(s_{t+1})$ and $\sum_{i=0}^{n-1} \beta(i) = 1$. 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:

$$\mathcal{T}^{\pi}_{\beta} v = r + \gamma \Bigg((1-\beta)\, P v + \beta\, (1-\lambda) \sum_{i=1}^{\infty} \lambda^{i-1} \widetilde{P}^{i} v \Bigg), \tag{4.8}$$

and a similar argument as Theorem 1 can be used to show the contraction property. The bias of exponential smoothing in policy evaluation can be characterized as:

$$\left\| v^{\pi} - v^{\pi}_{\beta} \right\|_{\infty} \leq \sum_{i=0}^{\infty} \gamma^{i} \left\| (P^{i} - ((1-\beta)P + \beta(1-\lambda) \sum_{j=1}^{\infty} \lambda^{j-1} \widetilde{P}^{j})^{i}) r \right\|_{\infty} . \tag{4.9}$$

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 using trend adjusted model Gardner Jr, 1985. An example of a temporal policy evaluation with temporal regularization using exponential smoothing is provided in Algorithm 6.

Algorithm 6 Policy evaluation with exponential smoothing

```
Input: $\pi, \alpha, \gamma, \beta, \lambda$
$p = V(s)$
for all steps do
    Choose $a \sim \pi(s)$
    Take action $a$, observe reward $r(s)$ and next state $s'$
    $V(s) = V(s) + \alpha(r(s) + \gamma((1-\beta)V(s') + \beta p))$
    $p = (1-\lambda)V(s) + \lambda p$
end for
```

4.1.3 Control case:

Temporal regularization can be extended to MDPs with actions by modifying the target of the value function (or the Q values) using temporal regularization. Experiments (Sec. 4.2.6) 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.

4.1.4 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 algorithm 7.

Algorithm 7 Temporally regularized semi-gradient TD

```
Input: policy $\pi$,$\beta$,$\gamma$
for all steps do
    Choose $a \sim \pi(s_t)$
    Take action $a$, observe $r(s)$, $s_{t+1}$
    $\theta = \theta + \alpha(r + \gamma((1-\beta)V_\theta(s_{t+1}) + \beta V_\theta(s_{t-1})) - V_\theta(s_t))\nabla V_\theta(s_t)$
end for
```

4.2 Experiment

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

4.2.1 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

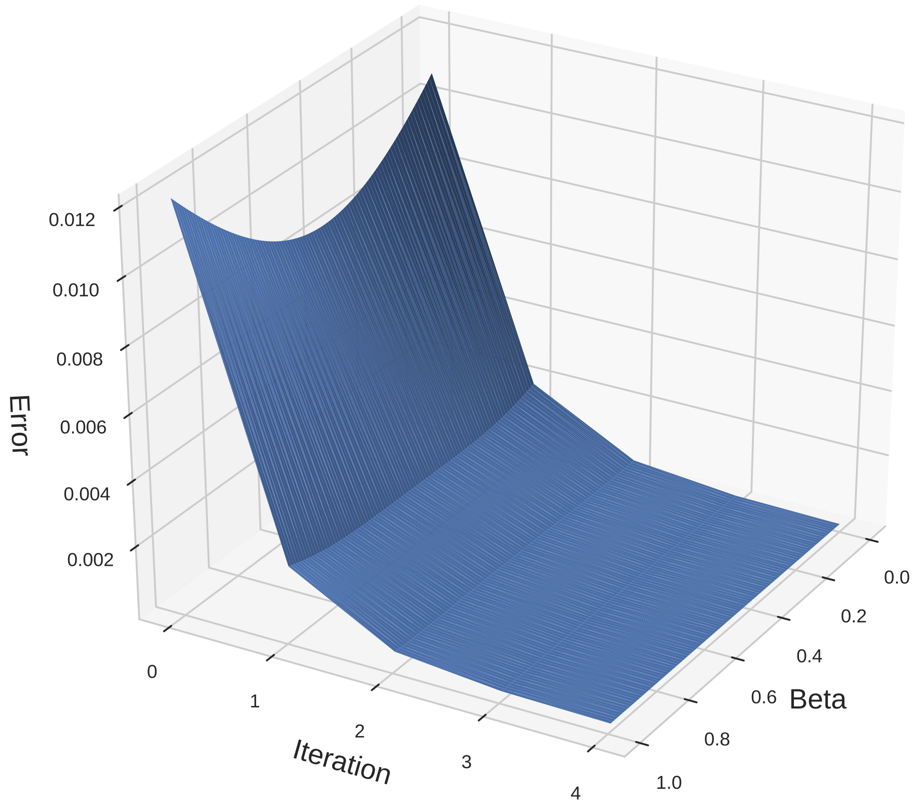


Figure 4.1: Distance between the stationary transition probabilities and the estimated transition probability for different values of regularization parameter β.

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 4.1 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.

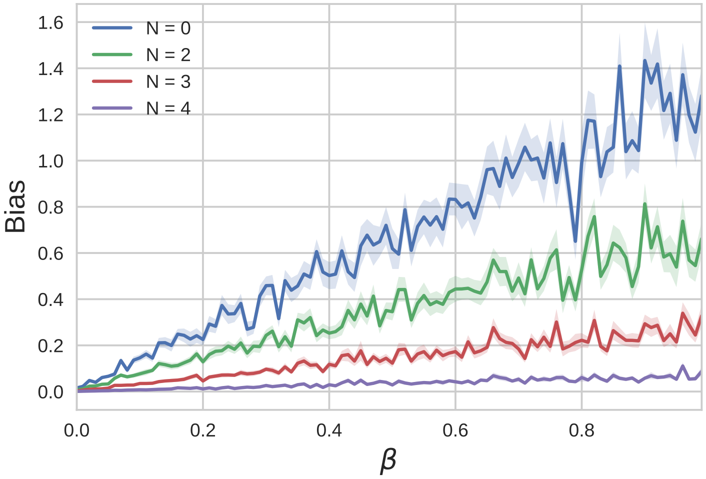


Figure 4.2: Mean difference between V^{π}_{β} and V^{π} given the regularization parameter β, for different amount of smoothed states N.

4.2.2 Bias

It is well known that reducing variance comes at the expense of inducing (smaller) bias. This has been characterized previously (Sec. **??**) in terms of the difference 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 4.2 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.

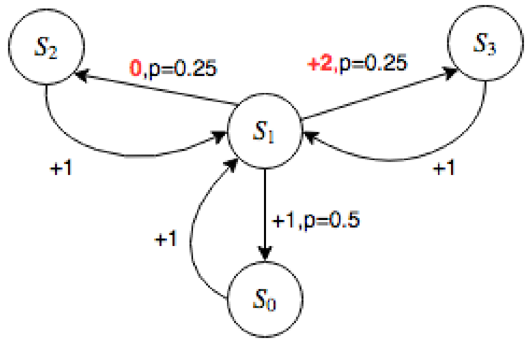


Figure 4.3: Synthetic MDP where state S_1 has high variance.

4.2.3 Variance

The primary motivation of this work is to reduce variance, therefore we now consider an experiment targeting this aspect. Figure 4.3 shows an example of a synthetic, 3-state 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 $V^\pi_\beta(S_1)$, $V^\pi(S_1)$, respectively.

Figure 4.4 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.

4.2.4 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 4.5) 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

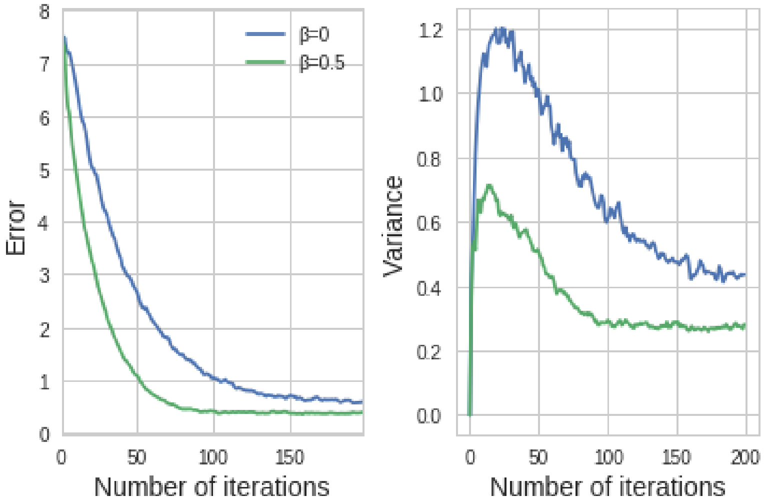


Figure 4.4: Left plot shows absolute difference between original ($V^{\pi}(S_1)$) and regularized ($V^{\pi}_{\beta}(S_1)$) state value estimates to the optimal value $V^{*}(S_1)$. Right plot shows the variance of the estimates V.

end).

Figure 4.5 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. It is also possible to consider more complex and powerful regularizers. Let us study a different time series prediction model, namely exponential averaging, as defined in (4.8). The complexity of such models is usually articulated by hyper-parameters, allowing complex models to improve performance by better adapting to problems. We illustrate this by comparing the performance of regularization using the previous state and an exponential averaging of all previous states. Fig. 4.6 shows the average error

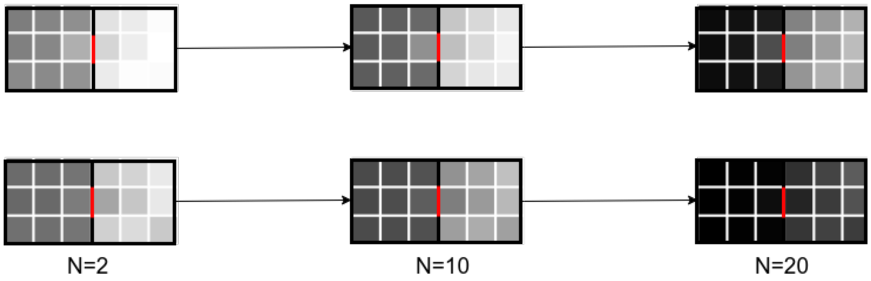


Figure 4.5: Proximity of the estimated state value to the optimal value after N trajectories. Top row is the original room environment and bottom row is the regularized one ($\beta = 0.5$). Darker is better.

on the value estimate using past state smoothing, exponential smoothing, and without smoothing. In this setting, exponential smoothing transfers information faster, thus enabling faster convergence to the true value.

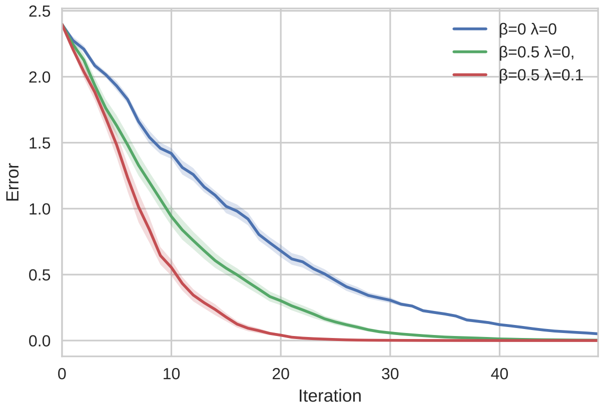


Figure 4.6: Benefits of complex regularizers on the room domain.

4.2.5 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. A learner evolving in this environment walks randomly along this line with a discount factor $\gamma = 0.95$. 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 $a_t \sim \mathcal{N}(0, 0.05)$. The state of the agent corresponds to the position perturbed by a zero-centered Gaussian noise ϵ_t, 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 1000 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. **??**. Figure 4.7 shows these errors, averaged over 1000 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. Using more complex regularizer can improve performance as shown in the previous section but this potential gain comes at the price of a potential loss in case of model misfit. Fig. 4.8 shows the absolute distance from the regularized state estimate (using exponential smoothing) to the optimal value while varying λ (higher λ = more smoothing). Increasing smoothing improves performance up to some point, but when λ is not well fit the bias becomes too strong and performance declines. This is a classic bias-variance tradeoff. 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.

4.2.6 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. PPO uses the general advantage estimator $\hat{A}_t = \delta_t + \gamma\lambda\delta_{t+1} + ... + (\gamma\lambda)^{T-t+1}\delta_T$ where $\delta_t = r_t + \gamma v(s_{t+1}) - v(s_t)$.

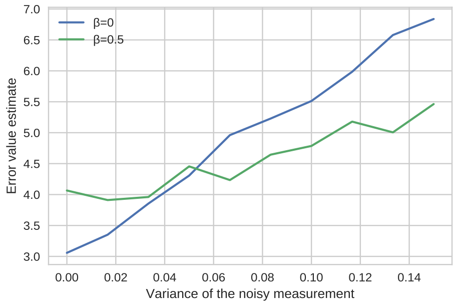


Figure 4.7: Absolute distance from the original (θ^{π}) and the regularized (θ^{π}_{β}) state value estimates to the optimal parameter θ^{*} given the noise variance σ^2 in state sensors.

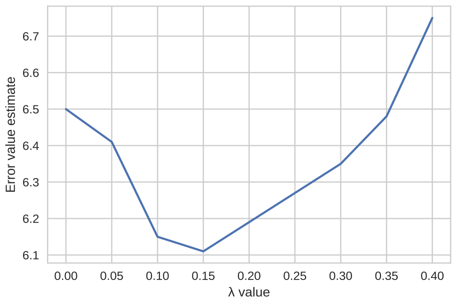


Figure 4.8: Impact of complex regularizer parameterization (λ) on the noisy walk using exponential smoothing.

We regularize δ_t such that $\delta_t^{\beta} = r_t + \gamma((1-\beta)v(s_{t+1}) + \beta\widetilde{v}(s_{t-1}))) - v(s_t)$ using exponential smoothing $\widetilde{v}(s_t) = (1-\lambda)v(s_t) + \lambda\widetilde{v}(s_{t-1})$ as described in Eq. (4.8). $\widetilde{v}$ is an exponentially decaying sum over all t previous state values encountered in the trajectory. We evaluate the performance in the Arcade Learning Environment Bellemare et al., 2013, where we consider the following performance measure:

$$\frac{\text{regularized} - \text{baseline}}{\text{baseline} - \text{random}}. \tag{4.10}$$

The hyper-parameters for the temporal regularization are $\beta = \lambda = 0.2$ and a decay of $1\mathrm{e}^{-5}$. 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.

The results reported in Figure 4.9 show 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 ap-

[1] The code can be found https://github.com/pierthodo/temporal_regularization.

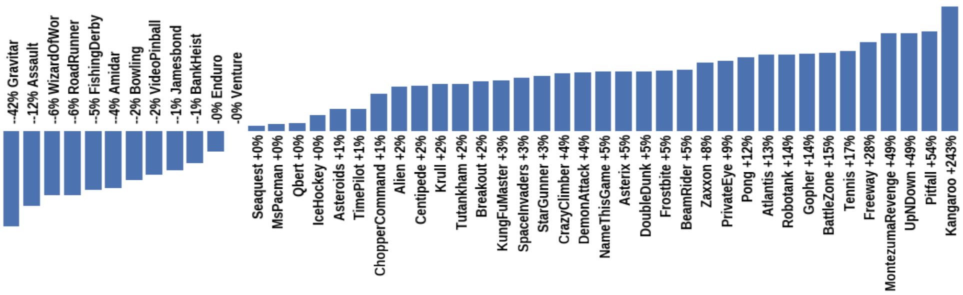


Figure 4.9: Performance (Eq. 4.10) of a temporally regularized PPO on a suite of Atari games.

proximation with deep learning. Also, as shown in previous experiments (Sec. 4.2.5), 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).

The full performance for each game can be found in figure 4.10.

4.3 Discussion

4.3.1 Noisy states:

Is is often assumed that the full state can be determined, while in practice, the Markov property rarely holds. This is the case, for example, when taking the four last frames to represent the state in Atari games Mnih, Kavukcuoglu, Silver, Rusu, et al., 2015.

A problem that arises when treating a partially observable MDP (POMDP) as a fully observable is that it may no longer be possible to assume that the value function is smooth over the state space Singh, Jaakkola, and Jordan, 1994. For example, the observed features may be similar for two states that are intrinsically different, leading to highly different values for states that are nearby in the state space. Previous experiments on noisy state representation (Sec. 4.2.5) and on the Atari games (Sec. 4.2.6) show that temporal regularization provides robustness to those cases. This makes it an appealing technique in real-world environments, where

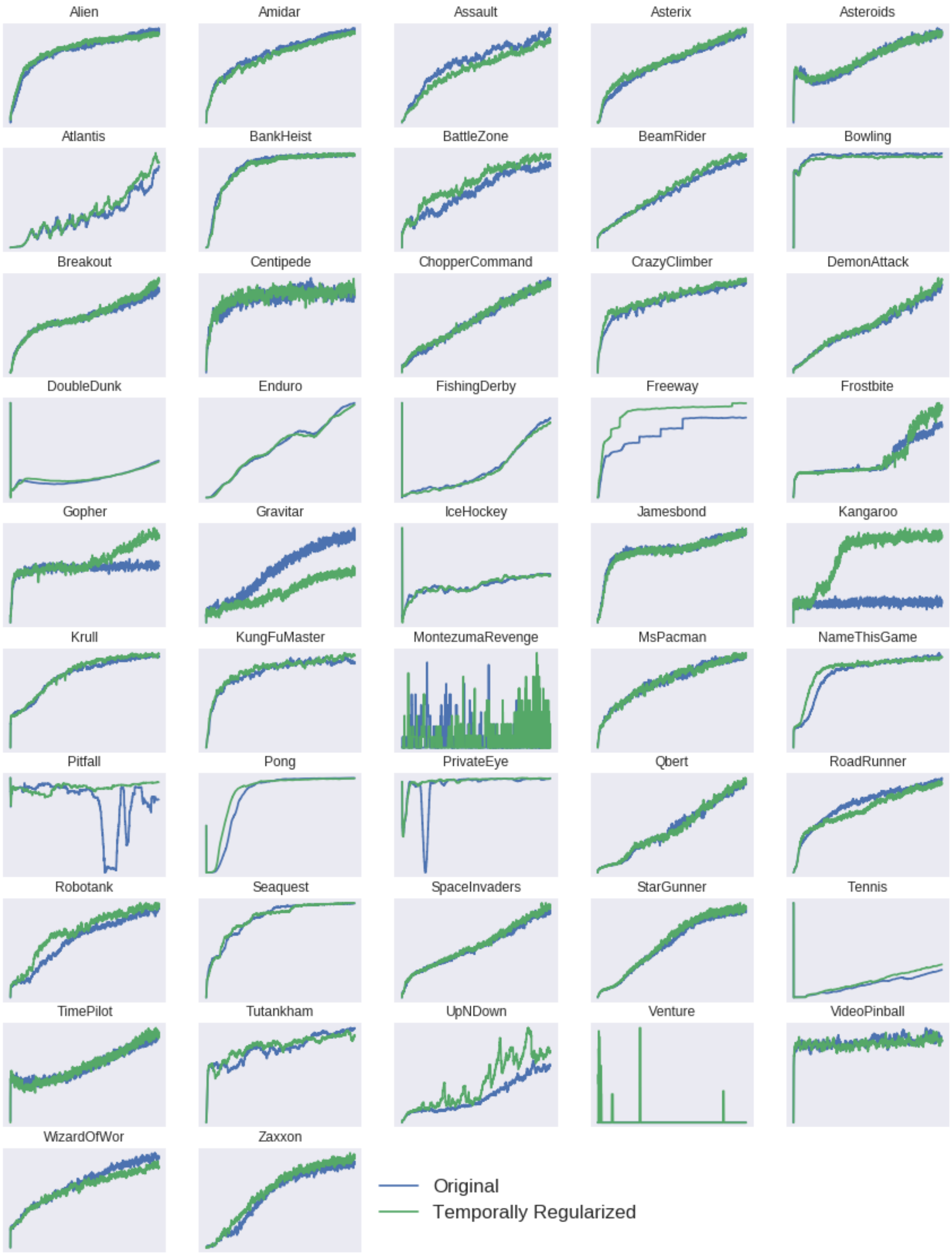


Figure 4.10: Average reward per episode on Atari games.

it is harder to provide the agent with the full state.

4.3.2 Choice of the regularization parameter:

The bias induced by the regularization parameter β can be detrimental for the learning in the long run. A first attempt to mitigate this bias is just to decay the regularization as learning advances, as it is done in the deep learning experiment (Sec. 4.2.6). Among different avenues that could be explored, an interesting one could be to aim for a state dependent regularization. For example, in the tabular case, one could consider β as a function of the number of visits to a particular state.

4.3.3 Smoother objective:

Previous work Laroche, 2018 looked at how the smoothness of the objective function relates to the convergence speed of RL algorithms. An analogy can be drawn with convex optimization where the rate of convergence is dependent on the Lipschitz (smoothness) constant **boyd2004convex**. By smoothing the value temporally we argue that the optimal value function can be smoother. This would be beneficial in high-dimensional state space where the use of deep neural network is required. This could explain the performance displayed using temporal regularization on Atari games (Sec. 4.2.6). The notion of temporal regularization is also behind multi-step methods R. S. Sutton and Barto, 1998; it may be worthwhile to further explore how these methods are related.

4.3.4 Policy Gradient 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

$$R(\pi) = \beta(\pi(a_{t-1}|s_{t-1}) - \pi(a_t|s_t)) \tag{4.11}$$

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 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.

5

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.

The problem of disentangling signal from noise in sequential domains is not specific to Reinforcement Learning and has been extensively studied in the Supervised Learning literature. In this work, we leverage ideas from time series literature and Recurrent Neural Networks to address the robustness of value functions in Reinforcement Learning. We propose Recurrent Value Functions (RVFs): an exponential smoothing of the value function. The value function of the current state is defined as an exponential smoothing of the values of states visited along the trajectory where the value function of past states are summarized by the previous RVF.

However, exponential smoothing along the trajectory can result in a bias when the value function changes dramatically through the trajectory (non-stationarity). This bias could be a problem if the environment encounters sharp changes, such as falling of a cliff, and the estimates are heavily smoothed. To alleviate this issue, we propose to use exponential smoothing on value functions using a trainable state-dependent emphasis function which controls the smoothing coefficients. Intuitively, the emphasis function adapts the amount of emphasis required on the current value function and the past RVF to reduce bias with respect to the optimal value estimate.

In other words, the emphasis function identifies important states in the environment. An important state can be defined as one where *its value differs significantly from the previous values along the trajectory.* For example, when falling off a cliff, the value estimate changes dramatically, making states around the cliff more salient. This emphasis function serves a similar purpose to a gating mechanism in a Long Short Term Memory cell of a Recurrent Neural Network Hochreiter and Schmidhuber, 1997.

To summarize the contributions of this work, we introduce RVFs to estimate the value function of a state by exponentially smoothing the value estimates along the trajectory. RVF formulation leads to a natural way of learning an emphasis function which mitigates the bias induced by smoothing. We provide an asymptotic convergence proof in tabular settings by leveraging the literature on asynchronous stochastic approximation Tsitsiklis, 1994. Finally, we perform a set of experiments to demonstrate the robustness of RVFs with respect to noise in continuous control tasks and provide a qualitative analysis of the learned emphasis function which provides interpretable insights into the structure of the solution.

5.1 Recurrent Value Functions (RVFs)

As mentioned earlier, performance of value-based methods are often heavily impacted by the quality of the data obtained Fox, Pakman, and Tishby, 2015; Pendrith, n.d. For example, in robotics, noisy sensors are common and can significantly hinder performance of popular methods Romoff et al., 2018. In this work, we propose a method to improve the robustness of value functions by estimating the value of a state s_t using the estimate at time step t and the estimates of previously visited states s_i where $i < t$. Mathematically, the Recurrent Value Function (RVF) of a state s at time step t is given by:

$$V^{\beta}(s_t) = \beta(s_t)V(s_t) + (1 - \beta(s_t))V^{\beta}(s_{t-1}), \tag{5.1}$$

where $\beta(s_t) \in [0,1]$. V^β estimates the value of a state s_t as a convex combination of current estimate $V(s_t)$ and previous estimate $V^\beta(s_{t-1})$. $V^\beta(s_{t-1})$ can be recursively expanded further, hence the name Recurrent Value Function. β is the emphasis function which updates the recurrent value estimate.

In contrast to traditional methods that attempt to minimize Eq. 3.13, the goal here is to find a set of parameters θ, ω that minimize the following error:

$$\begin{aligned} \mathcal{L}(\theta,\omega) &= \mathop{\mathbb{E}}_{\pi}[(V^\pi - V^\beta_\theta)^2], \\ V^\beta_\theta(s_t) &= \beta_\omega(s_t)V(s_t) + (1-\beta_\omega(s_t))(V^\beta_\theta(s_{t-1})), \end{aligned} \tag{5.2}$$

where V is a function parametrized by θ, and β_ω is a function parametrized by ω. This error is similar to the traditional error in Eq. 3.13, but we replace the value function with V^β_θ. In practice, V^π can be any target such as TD(0), TD(N), TD(λ) or Monte Carlo R. S. Sutton and Barto, 1998 which is used in Reinforcement Learning. We minimize Eq. 5.2 by updating θ and ω using the semi-gradient technique which results in the following update rule:

$$\begin{aligned} \theta &= \theta + \alpha\delta_t\nabla_\theta V^\beta_\theta(s_t), \\ \omega &= \omega + \alpha\delta_t\nabla_\omega V^\beta_\theta(s_t), \end{aligned} \tag{5.3}$$

where $\delta_t = V^\pi(s_t) - V^\beta_\theta(s_t)$ is the TD error with RVF in the place of the usual value function. The complete algorithm using the above update rules can be found in Algorithm 8.

Algorithm 8 Recurrent Temporal Difference(0)

1.2

Input: π,γ,θ,ω Initialize: $V^\beta_\theta(s_0) = V(s_0)$ Output: θ, ω Return the learned parameters t Take action $a \sim \pi(s_t)$, observe $r(s_t), s_{t+1}$ $V^\beta_\theta(s_t) = \beta_\omega(s_t)V(s_t) + (1 - \beta_\omega(s_t))V^\beta_\theta(s_{t-1})$ Compute the RVF $\delta_t = V^\pi(s_t) - V^\beta_\theta(s_t)$ Compute TD error with respect to RVF $\theta = \theta + \alpha\delta_t\nabla_\theta V^\beta_\theta(s_t)$ Update parameters of the value function θ $\omega = \omega + \alpha\delta_t\nabla_\omega V^\beta_\theta(s_t)$ Update parameters of the emphasis function ω

As discussed earlier, β_ω learns to identify states whose value significantly differs from previous estimates. While optimizing for the loss function described in Eq. 5.2, the $\beta_\omega(s_t)$ learns to bring the RVF V_θ^β closer to the target V^π. It does so by placing greater emphasis on whichever is closer to the target, either $V(s_t)$ or $V_\theta^\beta(s_{t-1})$. Concisely, the updated behaviour can be split into four scenarios. A detailed description of these behaviours is provided in Table 5.1. Intuitively, if the past is not aligned with the future, β will emphasize the present. Likewise, if the past is aligned with the future, then β will place less emphasis on the present. This behaviour is further explored in the experimental section.

Table 5.1: Behaviour of β based on the loss

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

Note that, the gradients of V_θ^β take a recursive form (gradient through time) as shown in Eq. 5.4. The gradient form is similar to LSTM Hochreiter and Schmidhuber, 1997, and GRU J. Chung et al., 2014 where β acts as a gating mechanism that controls the flow of gradient. LSTM uses a gated exponential smoothing function on the hidden representation to assign credit more effectively. In contrast, we propose to exponentially smooth the outputs (value functions) directly rather than the hidden state. This gradient can be estimated using backpropagation through time by recursively applying the chain rule where:

$$\nabla_\theta V_\theta^\beta(s_t) = \beta_\omega(s_t)\nabla_\theta V(s_t) + (1-\beta_\omega(s_t)).\nabla_\theta V_\theta^\beta(s_{t-1}) \tag{5.4}$$

However, this can become computationally expensive in environments with a large episodic length, such as continual learning. Therefore, we could approximate the gradient $\nabla_\theta V_\theta^\beta(s_t)$ using a recursive *eligibility* trace:.

$$e_t = \beta_\omega(s_t)\nabla_\theta V(s_t) + (1-\beta_\omega(s_t))e_{t-1}. \tag{5.5}$$

In the following section, we present the asymptotic convergence proof of RVF.

Asymptotic convergence

For this analysis, we consider the simplest case: a tabular setting with TD(0) and a fixed set of β. In the tabular setting, each component of θ and ω estimates one particular state, allowing us to simplify the notation. In this section, we simplify the notation by dropping θ and ω such that $V(s_t) = V(s_t)$ and $\beta_\omega(s_t) = \beta_t$. In the tabular setting, convergence to the fixed point of an operator is usually proven by casting the learning algorithm as a stochastic approximation Tsitsiklis, 1994; Borkar, 2009; Borkar and Meyn, 2000 of the form:

$$\theta_{t+1} = \theta_t + \alpha(\mathcal{T}\theta_t - \theta_t + w(t)), \tag{5.6}$$

where $\mathcal{T} : \mathbb{R}^{|\mathbb{S}|} \to \mathbb{R}^{|\mathbb{S}|}$ is a contraction operator and $w(t)$ is a noise term. The main idea is to cast the Recurrent Value Function 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 only updates certain states at each time step.

We consider the stochastic approximation formulation described in Eq. 5.6 with the following operator $\mathcal{T}^\beta : \mathbb{R}^{|\mathbb{S}|} \to \mathbb{R}^{|\mathbb{S}|}$ for any $i \leq t$:

$$\mathcal{T}^\beta V(s_i) = \mathop{\mathbb{E}}_{\pi}[r_t + \gamma V(s_{t+1}) + \Delta_t(s_i)] \tag{5.7}$$

for all states s_i with $\beta_i \in (0, 1]$. $\Delta_t(s_i)$ can be interpreted as a regularization term composed of the difference between $V(s_i)$ and $V^\beta(s_t)$.

To obtain this operator we first examine the update to $V(s_i)$ made during the trajectory at time step t:

$$\begin{aligned} V(s_i) &= V(s_i) + \alpha e_t(s_i)(r_t + \gamma V(s_{t+1}) - V^\beta(s_t)) \\ &= V(s_i) + \alpha e_t(s_i)(r_t + \gamma V(s_{t+1}) + \Delta_t(s_i) - V(s_i)) \end{aligned} \tag{5.8}$$

where $\Delta_t(s_i) = (1 - C_t(s_i))(V(s_i) - \widetilde{V}_t(s_i))$ and $C_t(s_i) = \beta_i \prod_{p=i+1}^{t}(1 - \beta_p)$. $\widetilde{V}_t(s_t)$ is a convex combination of all V encountered in the trajectory, with the exception of $V(s_i)$, weighted by their respective contribution(β) to the estimate $V^{\beta}(s_t)$. For example, if we consider updating $V(s_2)$ 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 $V(s_1)$. The main component of the error will be $r_t + \gamma V(s_4) - V(s_1)$. An example on how to obtain this decomposition can be found in the section 5.4.1.1 of Appendix. 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 to the learning. Bounding the magnitude of Δ to ensure contraction is the key concept used in this paper to ensure asymptotic convergence.

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 an infinite number of 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 minimum reward, and 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 an MDP. All rewards are assumed to be positive and scaled in the range $[R_{\min}, \widetilde{R}_{\max}]$ such that the scaled maximum reward $\widetilde{R}_{\max}$ satisfies the following:*

$$D\widetilde{R}_{max} \leq R_{min} \qquad D > \gamma \tag{5.9}$$

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

In theory, scaling the reward is reasonable as it does not change the optimal solution of the MDP Hasselt et al., 2016. In practice, however, this may be constraining as the range of the reward may not be known beforehand. This assumption could be relaxed by considering the trajectory's information to bound Δ. As an example, one could consider any physical system where transitions in the state space are smooth (continuous state space) and bounded by some Lipschitz constant in a similar manner than Shah and Xie, 2018.

As mentioned earlier, the key component of the proof is to control the magnitude of the term in Eq. 5.8: $\Delta_t(s_i) = (1 - C_t(s_i))(V(s_i) - \widetilde{V}_t(s_i))$. 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* $V_{\max} = \frac{\widetilde{R}_{\max}}{1-(\gamma+(1-D))}$ *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 functions such that $\forall s \in \mathbb{S} \quad V_{\min} \leq V(s) \leq V_{\max}$. The functions V are initialized in X.*
- *For a given D and γ we select C such that $\Delta \leq (1-C)(V_{\max} - V_{\min}) \leq (1 - D)V_{\min}$.*

We outline important details of the proof here. A full version can be found in Appendix 5.4.1.2. For two sets of value functions $U, V \in X$:

$$\max_s \mathop{\mathbb{E}}_{\pi}[\Delta^V(s) - \Delta^U(s))] \leq \max_s \mathop{\mathbb{E}}_{\pi}[(1-D)(V(s) - U(s))], \tag{5.10}$$

where Δ^U is the Δ described in Eq. 5.8 with value function U. For any γ contraction operator, we can now guarantee that $\mathcal{T}^\beta$ is a $\gamma + (1-D)$ contraction operator, where $\gamma + (1-D) < 1$ holds from Assumption 3. We provide an example in Appendix 5.4.1.3 to set C based on γ and D. We can guarantee that V converges to a fixed point of the operator $\mathcal{T}^\beta$ with probability $= 1$ using Theorem 3 of Tsitsiklis, 1994. The assumptions of Theorem 3 of Tsitsiklis, 1994 are discussed in section 5.4.1.4 of Appendix.

5.2 Related work

One important similarity of RVFs is with respect to the online implementation of λ return R. S. Sutton and Barto, 1998; Dayan, 1992. Both RVF and online λ returns have an eligibility trace form, but the difference is in RVF's capacity to ignore a state based on β. In this paper we argue that this can provide more robustness to noise and partial observability. The ability of RVF to emphasize a state is similar to the interest function in emphatic TD Mahmood et al., 2015, however, learning a state-dependant interest function and λ remains an open problem. In contrast, RVF has a natural way of learning β by comparing the past and the future. The capacity to ignore states shares some motivations to semi-Markov decision process Puterman, 1990. Learning β and ignoring states can be interpreted as learning temporal abstraction over the trajectory in policy evaluation. In reward shaping literature, several works such as Temporal Value Transport Hung et al., 2018, Temporal Regularization Thodoroff et al., 2018, Natural Value Approximator Xu et al., 2017 attempt to modify the target to either enforce temporal consistency or to assign credit efficiently. This departs from our formulation as we consider estimating a value function directly by using the previous estimates rather than by modifying the target. As a result of modifying the estimate, RVFs can choose to ignore a gradient while updating, which is not possible in other works. For example, in settings where the capacity is limited, updating on noisy states can be detrimental for learning. Finally, RVF can also be considered as a partially observable method Kaelbling, Littman, and Cassandra, 1998. However, it differs significantly from the literature as it does not attempt to infer the underlying hidden state explicitly, but rather only decides if the past estimates align with the target. We argue that inferring an underlying state may be significantly harder than learning to ignore or emphasize a state based on its value. This is illustrated in the next section.

5.3 Experiments

In this section, we perform experiments on various tasks to demonstrate the effectiveness of RVF. First, we explore RVF robustness to partial observability on a synthetic domain. We then showcase RVF's robustness to noise on several complex continuous control tasks from the Mujoco suite Todorov, Erez, and Tassa, 2012. An example of policy evaluation is also provided in Appendix 5.4.3.1 as a reading.

5.3.1 Partially observable multi-chain domain

We consider the simple chain MDP described in Figure 5.1. This MDP has three chains connected together to form a Y. Each of the three 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 up or down. The chain on top receives a reward of $+1$ while the one at the bottom receives a reward of -1. Every other transition has a reward of 0, unless specified otherwise. We explore the capacity of recurrent learning to solve a partially observable task in the Y chain. In particular, we consider the case where some states are aliased (share a common representation). The representation of the states S_4 and S_5 in Figure 5.1 are aliased. The goal of this environment is to correctly estimate the value of the aliased state $V^{\pi}(S_4) = 0.9, V^{\pi}(S_5) = -0.9$ (due to the discount factor(0.9) and the length of each chain being 3). When TD methods such as TD(0) or TD(λ) are used, the values of the aliased states S_4 and S_5 are close to 0 as the reward at the end of the chain is $+1$ and -1. However, when learning β (emphasis function β is modelled using a sigmoid function), Recurrent Value Functions achieve almost no error in their estimate of the aliased states as illustrated in Figure 5.2. This can be explained by observing that $\beta \rightarrow 0$ on the aliased state due to the fact that the previous values along the trajectory are better estimates of the future than those of the aliased state. As $\beta \rightarrow 0$, $V_{\theta}^{\beta}(S_4)$ and $V_{\theta}^{\beta}(S_5)$ tend to rely on their more accurate previous estimates,

[b].45

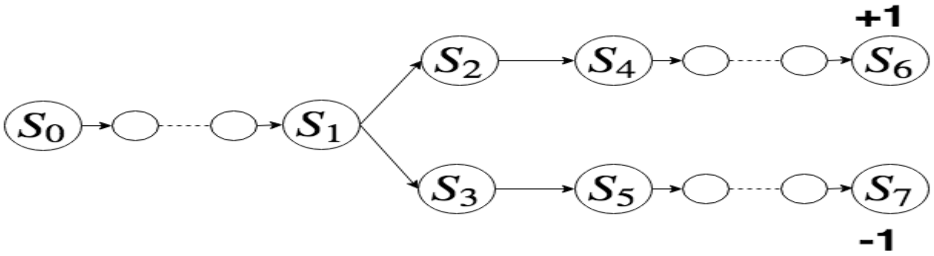


Figure 5.1: Simple chain MDP.

[b].45

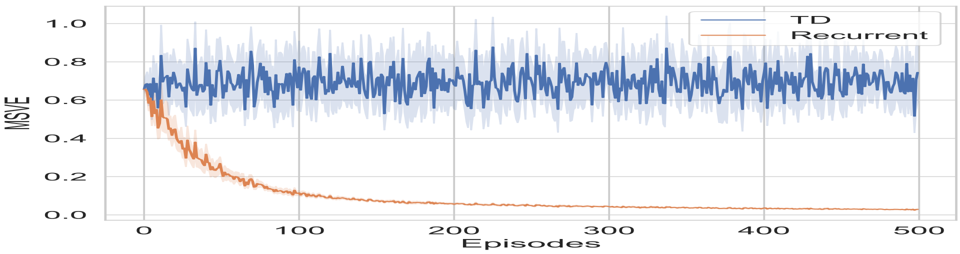


Figure 5.2: Results on the aliased Y-chain.

Figure 5.3: (a) Simple chain MDP: The agent starts at state S_0 and navigates along the chain. States S_4 and S_5 are aliased. (b) Results on the aliased Y-chain on various methods, such as TD(0), TD(λ), GRU, RTD(0), and Optimal RTD(0) (O-RTD(0)) averaged over 20 random seeds.

$V_\theta^\beta(S_2)$ and $V_\theta^\beta(S_3)$. We see that learning to ignore certain states can at times be sufficient to solve an aliased task. We also compare with a recurrent version (O-RTD) where optimal values of β are used. In this setting, $\beta(S_1) = \beta(S_2) = \beta(s_3) = 1$ and other states have $\beta = 0$. Another interesting observation is with respect to Recurrent Neural Networks. RNNs are known to solve tasks which have partial observability by inferring the underlying state. LSTM and GRU have many parameters that are used to infer the hidden state. Correctly learning to keep the hidden state intact can be sample-inefficient. In comparison, β can estimate whether or not to put emphasis (confidence) on a state value using a single parameter. This is illustrated in Figure 5.2 where RNNs take 10 times more episodes to learn the optimal value when compared to RVF. This illustrates a case where learning to ignore a state is easier than inferring its hidden representation. The results displayed in Figure 5.2 are averaged over 20 random seeds. For every method, a hyperparameter search is done to obtain their optimal value. These can be found in Appendix 5.4.3.2. We noticed that the emphasis function is easier to learn if the horizon of the target is longer, since a longer horizon provides a better prediction of the future. To account for this, we use λ-return as a target.

5.3.2 Deep Reinforcement Learning

Next, we test RVF on several environments of the Mujoco suite Todorov, Erez, and Tassa, 2012. We also evaluate the robustness of different algorithms by adding ϵ sensor noise (drawn from a normal distribution $\epsilon \sim N(0,1)$) to the observations as presented in Zhang, Ballas, and Pineau, 2018. We modify the critic of A2C Wu et al., 2017 (R-A2C) and Proximal Policy Optimization (R-PPO) Schulman, Wolski, et al., 2017 to estimate the recurrent value function parametrized by θ. We parametrize β using a seperate network with the same architecture as the value function (parametrized by ω). We minimize the loss mentioned in Eq. 5.2 but replace the target with generalized advantage function (V^λ) Schulman, Moritz, et al., 2015 for PPO and TD(n) for A2C. Using an automatic differentiation library (Pytorch Paszke et al., 2017), we differentiate the loss through the modified estimate to learn θ and ω. The default optimal hyperparameters of PPO and A2C are used. Due to the batch nature of PPO, obtaining the trajectory information to create the computational graph can be costly. In this regard, we cut the backpropagation after N timesteps in a similar manner to truncated backpropagation through time. The number of backpropagation steps is obtained using hyperparameter search. Details can be found in Appendix 5.4.3.3. We use a truncated backprop of $N = 5$ in our experiments as we found no empirical improvements for $N = 10$. For a fairer comparison in the noisy case, we also compare the performance of two versions of PPO with an LSTM. The first version processes one trajectory every update. The second uses a buffer in a similar manner to PPO, but the gradient is cut after 5 steps as the computation overhead from building the graph every time is too large. The performance reported is averaged over 20 different random seeds with a confidence interval of 68% displayed [1]

5.3.2.1 Performance

./fig/perf_mujoco.pdf

Figure 5.4: Performance on Mujoco tasks. Results on the first row are generated without noise and on the second row by inducing a Gaussian noise ($\epsilon \sim \mathbb{N}(0, 0.1)$) in the sensor inputs.

As demonstrated in Figure 5.4, we observe a marginal increase in performance on several tasks such as Swimmer, Walker, Hopper, Half Cheetah and Double Inverted Pendulum in the fully observable setting. However, severe drops in performance were observed in the vanilla PPO when we induced partial observability by introducing a Gaussian noise to the observations. On the other hand, R-PPO (PPO with RVF) was found to be robust to the noise, achieving significantly higher performance in all the tasks considered. In both cases, R-PPO outperforms the partially observable models (LSTM). The mean and standard deviation of the emphasis function for both noiseless and noisy versions can be found in Appendix(5.9, 5.10). At the same time, A2C

[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.

performance on both vanilla and recurrent versions (referred as R-A2C) were found to be poor. We increased the training steps on both versions and noticed the same observations as mentioned above once A2C started to learn the task. The performance plots, along with the mean and standard deviation of the emphasis function during training, can be found in Appendix (5.11, 5.12, 5.13, 5.14).

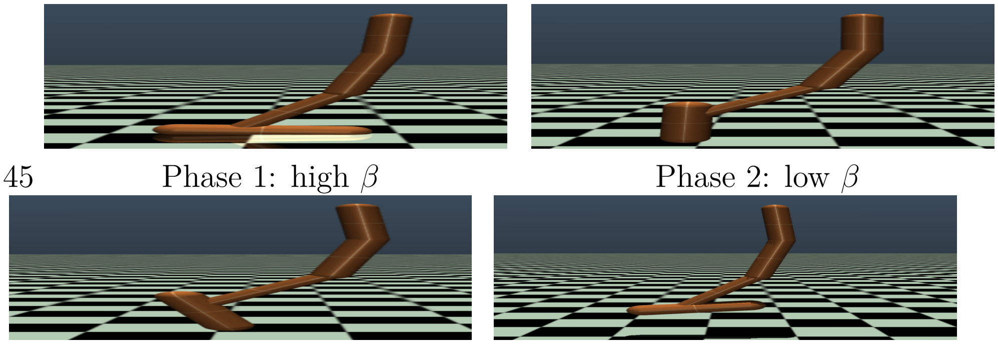
5.3.2.2 Qualitative interpretation of the emphasis function β

Hopper: At the end of training, we can qualitatively analyze the emphasis function (β) through the trajectory. We observe cyclical behaviour shown in Figure 5.6, where different colours describe various stages of the cycle. The emphasis function learned to identify *important states* and to ignore the others. One intuitive way to look at the emphasis function(β) is: *If I were to give a different value to a state, would that alter my policy significantly?* We observe an increase in the value of the emphasis function (β) when the agent must make an important decision, such as jumping or landing. We see a decrease in the value of the emphasis function (β) when the agent must perform a trivial action. This pattern is illustrated in Figure 5.5 and 5.6. This behaviour is cyclic and repetitive, a video of which can be found in the following link[2].

5.4 Discussions and Future Work

Temporal Credit assignment: As mentioned earlier, we can control the flow of gradient by using emphasis function $\beta_\omega(s_t)$ and pass gradient to the states that contributed to the reward but are located several time-steps earlier. We could potentially do credit assignment on states that are temporally far away by forcing the emphasis function between these states to be close to 0. This setting could be useful in problems with long horizons, such as lifelong learning and continual learning.

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



.45 Phase 1: high β

Phase 2: low β

Phase 3: high β

Phase 4: low β

Figure 5.5: Cyclical behaviour of β on Hopper.

.45

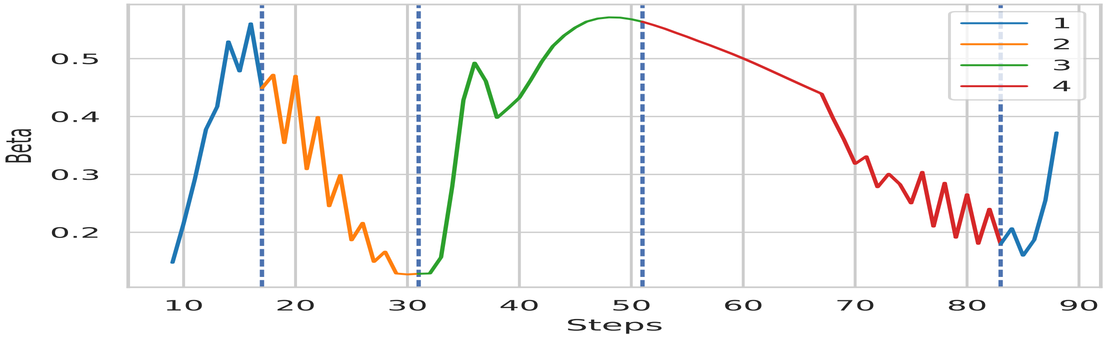


Figure 5.6: Behaviour of β through the trajectory.

Figure 5.7: (a) The emphasis function learns to emphasize key states in the environment. The emphasis function is high when the agent is making important decisions, such as landing or taking off (Phase 1 and Phase 3). The emphasis function is low when the agent is making decisions while it is in the air (Phase 2 and Phase 4). (b) Behaviour of the emphasis function along the trajectory for various phases described in (a) for one period. The emphasis function keeps repeating the behaviour.

β 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, and use β to set the λ to bootstrap from interesting states. Indeed, bootstrapping on states with a similar value (low β) than the one estimated will only result in variance. The most informative updates come from bootstrapping on states with different values (high β). We also believe β to be related to the concepts of bottleneck states Tishby and Polani, 2011 and reversibility.

Partially observable domain: As demonstrated earlier, RVFs are able to correctly estimate the value of an aliased/noisy state using the trajectory's estimate. We believe that this is a promising area to explore because, as the experiments suggest, ignoring an uninformative state can sometimes suffice to learn its value function. This is in contrast to traditional POMDP methods which attempt to infer the belief state. Smoothing the output with a gating mechanism could also be useful for sequential tasks in Supervised Learning, such as regression or classification.

Adjusting for the reward: In practice, some environments in Reinforcement Learning have a constant reward at every time step, potentially inducing bias in V_θ^β estimates. It would be possible to modify the RVF formulation to account for the reward that was just seen, such that $V_\theta^\beta(s_t) = \beta V(s_t) + (1-\beta)(V_\theta^\beta(s_{t-1}) - r_{t-1})$. Whether or not subtracting the reward can reduce the bias will depend on the environment considered.

Conclusion: In this work we propose Recurrent Value Functions to address variance issues in Model-free Reinforcement Learning. First, we prove the asymptotic convergence of the proposed method. We then demonstrate the robustness of RVF to noise and partial observability in a synthetic example and on several tasks from the Mujoco suite. Finally, we describe the behaviour of the emphasis function qualitatively.

5.4.1 Convergence Proof

TD(0) is known to converge to the fixed point of the bellman operator R. S. Sutton, 1988:

$$\mathcal{T}V(s_t) = \mathop{\mathbb{E}}_{s_{t+1}\sim\pi}[r(s_t) + \gamma V(s_{t+1})] \tag{5.11}$$

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

$$w(t) = r_t + \gamma V(s_{t+1}) - \mathop{\mathbb{E}}_{s_{t+1}\sim\pi}[r + \gamma V(s_{t+1})] \tag{5.12}$$

5.4.1.1 Derivation of $\widetilde{V}$, Eq. 5.8

We take an example with $t = 3$ and consider $i = 2$:

$$\begin{aligned} V_\theta^\beta(s_3) &= \beta_3 V(s_3) + (1-\beta_3)\beta_2 V(s_2) + (1-\beta_3)(1-\beta_2)V(s_1) \\ &= V(s_2) - (1-(1-\beta_3)\beta_2)(V(s_2) - \frac{\beta_3 V(s_3) + (1-\beta_3)(1-\beta_2)V(s_1)}{(1-(1-\beta_3)\beta_2)}) \\ &= V(s_2) - (1-(1-\beta_3)\beta_2)(V(s_2) - \widetilde{V}_t(s_i)) \end{aligned} \tag{5.13}$$

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:

$$\begin{aligned} &\frac{\beta_3 + (1-\beta_3)(1-\beta_2)}{(1-(1-\beta_3)\beta_2)} = 1 \\ &\equiv \beta_3 + (1-\beta_3)(1-\beta_2) = (1-(1-\beta_3)\beta_2) \\ &\equiv \beta_3 + (1-\beta_3)\beta_2 + (1-\beta_3)(1-\beta_2) = 1 \end{aligned} \tag{5.14}$$

where the last line is true because $\beta \in (0, 1]$

5.4.1.2 Proof theorem 1

Theorem 1. *Let's define* $V_{\max} = \frac{\widetilde{R}_{\max}}{1-(\gamma+(1-D))}$ *and* $V_{\min} = \frac{R_{\min}}{1-(\gamma-(1-D))}$. *If the following holds*

- *Let X be the set of V functions such that $\forall s \in \mathbb{S} \quad V_{\min} \leq V(s) \leq V_{\max}$. We assume the functions are initialized in X.*

- *For a given D and γ we select C such that $(1-C)(V_{\max} - V_{\min}) \leq (1-D)V_{\min}$*

then $\mathcal{T}^\beta : X \to X$ is a contractive operator.

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

$$\begin{aligned} \widetilde{\mathcal{T}^{\beta}}V(s) &\leq \widetilde{R}_{\max} + \gamma V_{\max} + (1-C)(V_{\max}-V_{\min}) \\ &\leq \widetilde{R}_{\max} + (\gamma + (1-D))V_{\max} \\ &\leq V_{\max} \end{aligned} \tag{5.15}$$

and

$$\begin{aligned} \widetilde{\mathcal{T}}^{\beta}V(s) &\geq R_{\min} + \gamma V_{\min} + (1-C)(V_{\min}-V_{\max}) \\ &\geq R_{\min} + (\gamma - (1-D))V_{\min} \\ &\geq V_{min} \end{aligned} \tag{5.16}$$

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

$$\begin{aligned} &\left\|\mathcal{T}^{\beta}V - \mathcal{T}^{\beta}U\right\|_{\infty} \\ &\leq \max_{s,s'} \mathop{\mathbb{E}}_{\pi}[\gamma V(s) + \Delta^{V}(s') - (\gamma U(s) + \Delta^{U}(s'))] \\ &\leq \max_{s,s'} \mathop{\mathbb{E}}_{\pi}[\gamma(V(s)-U(s)) + (1-D)(V(s')-U(s'))] \\ &\leq \max_{s} \mathop{\mathbb{E}}_{\pi}[((1-D)+\gamma)(V(s)-U(s))] \\ &\leq ((1-D)+\gamma)\left\|V-U\right\|_{\infty} \end{aligned} \tag{5.17}$$

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

5.4.1.3 Selecting C

To select C based on γ and D it suffice to solve analytically for:

$$\begin{aligned} &(1-C)(V_{\max}-V_{\min}) \leq (1-D)V_{\min} \\ &\equiv (1-C)\frac{\widetilde{R}_{\max}}{1-(\gamma+(1-D)} \leq ((1-D)+(1-C))\frac{R_{\min}}{1-(\gamma-(1-D)} \\ &\equiv \frac{(1-C)(1-(\gamma-(1-D)))}{(1-(\gamma+(1-D))((1-D)+(1-C)}\widetilde{R}_{\max} \leq R_{\min} \\ &\equiv \frac{D(1-C)(1-(\gamma-(1-D)))}{(1-(\gamma+(1-D))((1-D)+(1-C)}R_{\min} \leq R_{\min} \end{aligned} \tag{5.18}$$

which is satisfied only if:

$$\frac{D(1-C)(1-(\gamma-(1-D)))}{(1-(\gamma+(1-D))((1-D)+(1-C)} \leq 1 \tag{5.19}$$

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

5.4.1.4 Assumption asynchronous stochastic approximation

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 $C \in \mathbb{R}$:

$$\begin{aligned} \sum_{i=0}^{\infty} \alpha_t(s)e_t(s) = \infty \quad \text{w.p.1} \\ \sum_{i=0}^{\infty} (\alpha_t(s)e_t(s))^2 \leq C \end{aligned} \tag{5.20}$$

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.

5.4.2 Derivation of β update rule

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

$$\min \quad \frac{1}{2}(V^{\pi}(s_t) - V^{\beta}_{\theta,\omega}(s_t))^2 \tag{5.21}$$

$$\tag{5.22}$$

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

$$\frac{d}{d\omega_{s_t}} \quad \left(\frac{1}{2}(V^{\pi}(s_t) - V^{\beta}_{\theta,\omega}(s_t))^2\right) = (V^{\pi}(s_t) - V^{\beta}_{\theta,\omega}(s_t))\Big(\frac{d(V^{\pi}(s_t) - V^{\beta}_{\theta,\omega}(s_t))}{d\omega_{s_t}}\Big)\Big)) \quad \text{by chain rule} \tag{5.23}$$

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

Therefore,

$$\frac{d}{d\omega} \quad \left(\frac{1}{2}(V^{\pi}(s_t) - V^{\beta}_{\theta,\omega}(s_t))^2 + \lambda\sigma(\omega_{s_t})\right) = \tag{5.24}$$

$$(V^{\pi}(s_t) - V^{\beta}_{\theta,\omega}(s_t))(V(s_t) - V^{\beta}_{\theta,\omega}(s_{t-1}))\left(\sigma(\omega_{s_t})(1 - \sigma(\omega_{s_t}))\right) + \lambda\left(\sigma(\omega_{s_t})(1 - \sigma(\omega_{s_t}))\right) = \tag{5.25}$$

$$\left(\sigma(\omega_{s_t})(1 - \sigma(\omega_{s_t}))\right)\left((V^{\pi}(s_t) - V^{\beta}_{\theta,\omega}(s_t))(V(s_t) - V^{\beta}_{\theta,\omega}(s_{t-1}))\right) \tag{5.26}$$

Finally, the update rule is simply a gradient step using the above derivative.

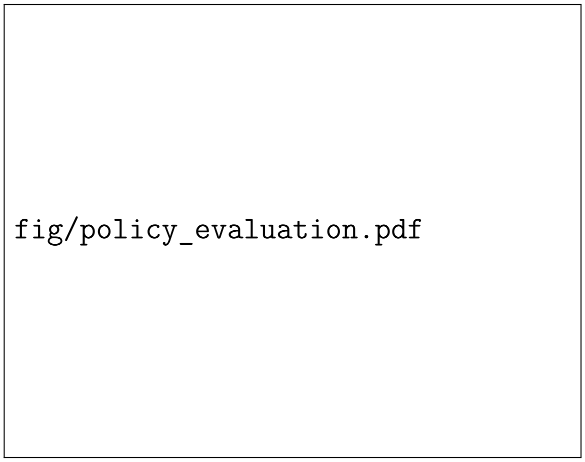


Figure 5.8: Comparison of RMSVE of various methods. Value Function estimated using RVF has lower error compared to Value Function estimated using TD and comparable to Eligibility traces

5.4.3 Experiment

5.4.3.1 Policy Evaluation

In this experiment, we perform a policy evaluation on CartPole task. In this environment the agent has to balance the pole on a cart. The agent can take one of the two actions available which would move the cart to either left or right. A reward of +1 is obtained for every time step and an episode is terminated if the cart move too far from the center or the pole dips below a certain angle.

In this task, we use a pretrained network to obtain the features that represent the underlying state. We train a linear function approximator using those features to estimate the value function. This experimental setup is similar to W. Chung et al., 2018 but the features in our case are obtained through a pretrained network instead of training a separate network. The samples are generated following a fixed policy and the same samples were used across all the methods to estimate value function. Each sample consists of 5000 transitions and the results are averaged over 40 such samples. We calculated the optimal value V_π using a Monte Carlo estimate for 2000 random states following the same policy. We use the trained linear network to predict value function on these 2000 states. Once we get the predictions we calculate their

Root Mean Square Value Error (RMSVE). The best hyperparameters were obtained through hyperparameter search for each method separately. The optimal learning rate was found to be 0.005 for TD and RVF while 0.0005 for eligibility traces when did a search in $\{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05\}$. The optimal beta learning rate was found to be 0.005 when we searched in $\{0.001, 0.0001, 0.01\}$ and the optimal lambda for eligibility traces was found to be 0.9 when searched in $\{0.1, 0.4, 0.7, 0.9\}$. The RMSVE on various methods such as TD, eligibility traces - online TD(λ) and the value functions of the state obtained using RVF algorithm are reported in Figure 5.8. We notice that the the value function learned through RVF algorithm has approximately the same error as the value function learned through eligibility traces. Both RVF and eligibility traces outperform TD methods.

5.4.3.2 Hyper-parameter TOY MDP

For every method the learning rate and λ was tuned for optimal performance in the range $[0, 1]$.

For RTD a learning rate of 0.5 for the value function and 1 for the beta function was found to be optimal with a lambda of 0.9.

For the GRU model we explored different amount of cell($\{1, 5, 10, 15, 20, 25\}$) to vary the capacity of the model. The optimal number of hidden cell we found is 10, learning rate 0.5 and lambda 0.9.

5.4.3.3 Deep Reinforcement Learning

The best hyperparameters were selected on 10 random seeds.

PPO: The following values were considered for the learning rate $\{3E-05, 6E-05, 9E-05, 3E-04, 6E-04\}$ and $N = \{2, 5, 10\}$. The optimal values for learning rate is the same one obtained in the original PPO paper $3E-4$ and $N = 5$. We also compare with a larger network for PPO to adjust for the additional parameter of β

the performance of vanilla PPO were found to be similar. In terms of computational cost, RVF introduce a computational overhead slowing down training by a factor of 2 on a CPU(Intel Skylake cores 2.4GHz, AVX512) compared to PPO. The results are reported on 20 new random seeds and a confidence interval of 68% is displayed.

A2C: The following values were considered for the learning rate $\{1E-05, 1E-04, 1E-03, 1E-02\}$ and the best learning rate was found to be $1E-04$. We tested bootstrapping on $5, 10, 15$ steps, we found no empirical improvements between 5 and 10 but noticed a significant drop in performance for bootstrapping > 10. We believe that, it is due to the increase in variance of the target as we increase the steps of the bootstrapping. The best hyperparameters were found by averaging across 10 random seeds. The results reported are averaged across 20 random seeds. The confidence interval of 95% is displayed.

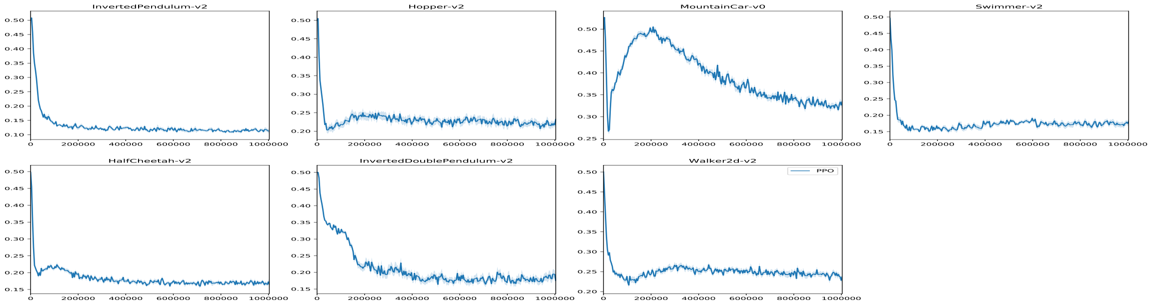


Figure 5.9: Mean beta values using recurrent PPO on Mujoco domains

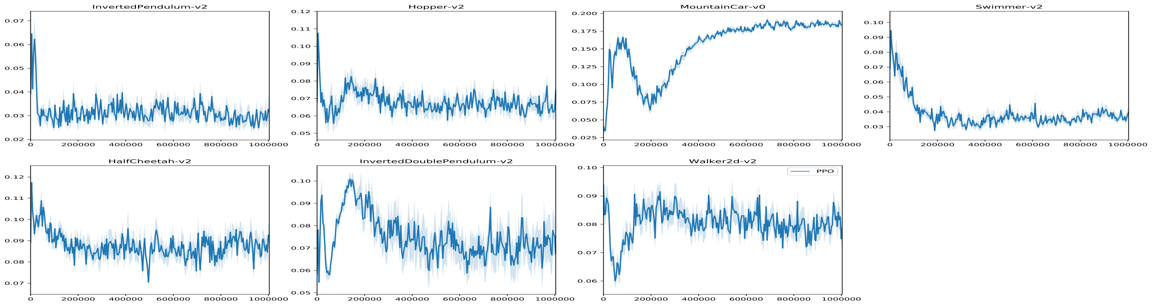


Figure 5.10: Standard deviation of beta using recurrent PPO on Mujoco domains

fig/Rewards_a2c_mujoco_sea_steps_5_noise_obs_0_0.pdf

Figure 5.11: Performance of a2c and recurrent a2c on Mujoco tasks without noise in observations for 10M steps

fig/Rewards_a2c_mujoco_sea_steps_5_noise_obs_0_1.pdf

Figure 5.12: Performance of a2c and recurrent a2c on Mujoco tasks with a Gaussian noise(0.1) in observations for 10M steps

fig/beta_mean_a2c_mujoco_sea_steps_5_noise_obs_0_1.pdf

Figure 5.13: The mean of the emphasis function on various Mujoco tasks with and without noise plotted against the number of updates

fig/beta_mean_a2c_mujoco_sea_steps_5_noise_obs_0_1.pdf

Figure 5.14: The standard deviation of the emphasis function on various Mujoco tasks with and without noise plotted against the number of updates

6

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.

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.

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*.

Chung, Wesley et al. (2018). "Two-Timescale Networks for Nonlinear Value Function Approximation". In:

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.

Fox, Roy, Ari Pakman, and Naftali Tishby (2015). “Taming the noise in reinforcement learning via soft updates”. In: *arXiv preprint arXiv:1512.08562*.

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*.

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

Hasselt, Hado P van et al. (2016). “Learning values across many orders of magnitude”. In: *Advances in Neural Information Processing Systems*, pp. 4287–4295.

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*.

Kaelbling, Leslie Pack, Michael L Littman, and Anthony R Cassandra (1998). “Planning and acting in partially observable stochastic domains”. In: *Artificial intelligence* 101.1-2, pp. 99–134.

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:

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.

Laroche, Van Seijen (2018). “IN REINFORCEMENT LEARNING, ALL OBJECTIVE FUNCTIONS ARE NOT EQUAL”. In: *ICLR Workshop*.

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*.

Pendrith, Mark D (n.d.). *On reinforcement learning of control actions in noisy and non-Markovian domains*. Citeseer.

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

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.

Romoff, Joshua et al. (2018). “Reward estimation for variance reduction in deep reinforcement learning”. In: *arXiv preprint arXiv:1805.03359*.

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, Tommi Jaakkola, and Michael I Jordan (1994). “Learning without state-estimation in partially observable Markovian decision processes”. In: *Machine Learning Proceedings 1994*. Elsevier, pp. 284–292.

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, 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 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.

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.

Zhang, Amy, Nicolas Ballas, and Joelle Pineau (2018). “A dissection of overfitting and generalization in continuous reinforcement learning”. In: *arXiv preprint arXiv:1806.07937*.