Stable Models and Temporal Difference Learning

Gaurav Manek

CMU-CS-23-103

April 2023

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
J. Zico Kolter, Chair
David Held
Deepak Pathak
Sergey Levine (UC Berkeley)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by Agency for Science, Technology and Research (A*STAR) Singapore and Robert Bosch GmbH

Keywords: Lyapunov Stability, Regularization, Deadly Triad, Offline Reinforcement Learning, Temporal Difference Learning, Reinforcement Learning, Neural Networks, Machine Learning, Artificial Intelligence

Abstract

In this thesis, we investigate two different aspects of stability: the stability of neural network dynamics models and the stability of reinforcement learning algorithms. In the first chapter, we propose a new method for learning Lyapunov-stable dynamics models that are stable by construction, even when randomly initialized. We demonstrate the effectiveness of this method on damped multi-link pendulums and show how it can be used to generate high-fidelity video textures.

In the second and third chapters, we focus on the stability of Reinforcement Learning (RL). In the second chapter, we demonstrate that regularization, a common approach to addressing instability, behaves counterintuitively in RL settings. Not only is it sometimes ineffective, but it can also cause instability. We demonstrate this phenomenon in both linear and neural network settings. Further, standard importance sampling methods are also vulnerable to this.

In the third chapter, we propose a mechanism to stabilize off-policy RL through resampling. Called Projected Off-Policy TD (POP-TD), it resamples TD updates to come from a convex subset of "safe" distributions instead of (as in other resampling methods) resampling to the on-policy distribution. We show how this approach can mitigate the distribution shift problem in offline RL on a task designed to maximize such shift.

Overall, this thesis advances novel methods for dynamics model stability and training stability in reinforcement learning, questions existing assumptions in the field, and points to promising directions for stability in model and reinforcement learning.

Acknowledgments

Although it is not possible to express my gratitude to everyone who has played a role in my academic and personal growth, I want to acknowledge and appreciate the countless friends, family, and collaborators whose contributions have been invaluable to me.

First and foremost, my advisor Zico Kolter. His guidance, advice, enthusiasm, expertise, encouragement, and unwavering support have been invaluable to me throughout this journey. His involvement has helped me develop a deeper understanding of the subject matter and how to approach research in a meaningful way. I also wish to thank Sergey Levine, David Held, and Deepak Pathak, for their invaluable feedback, suggestions, and guidance as members of my thesis committee.

I would like to express my sincere appreciation to my collaborators on the POP-TD project Melrose Roderick and Felix Berkenkamp, for their insight, good humor, time, patience, and tenacity.

From the department, I wish to thank Ruben Martins for engaging with my fledgeling ideas in his field. Also, the support staff who make the department what it is: Catherine Copetas, Deb Cavlovich, Ann Stetser, Charlotte Yano, and the many, many others. I also owe a debt of gratitude to Francisco Maturana, my roommate of six years, colleague in the CS department, and friend.

From A*STAR, I wish to thank the many people in the past and present who have molded me as researchers and supported me through my PhD, especially Vijay Chandrasekhar and Li Xiaoli. I also wish to thank the many senior researchers who have invested their time in me, and the dedicated scholarship officers who have supported me for a decade. From my undergraduate years in Brown I need to thank Shriram Krishnamurthi and Stefanie Tellex for starting me down this journey.

And most of all, I would like to express my heartfelt thanks to my parents Mukesh and Sonal Manek, my brother Gautam Manek, and my extended family for their unwavering love, encouragement, and support throughout the highs and lows of my academic journey. Their steadfast presence, undying love, and unwavering support have been the foundation upon which I have built my academic success, and I am forever indebted to them.

Contents

Introduction

In this thesis we examine two notions of stability: that of neural network dynamics models and the training of reinforcement learning algorithms. There is a natural transition from the first notion of stability to the second is natural: the parameters of a stably trained model circumscribes, in parameter-space, a stable trajectory. This relationship between stabilities has significant precedence in the foundational work of Temporal Difference (TD) learning theory [54].

In the first chapter we propose a new method for learning Lyapunov-stable dynamical models and the certifying Lyapunov function in a fully end-to-end manner. Instead of enforcing stability by some loss function, we guarantee stability everywhere by construction. This works by carefully constructing a neural network to act as a Lyapunov function, learning a separate, unconstrained dynamics model, and then combining these two models with a novel reprojection layer. This produces models that are guaranteed stable by construction everywhere in the state space, even without any training. We show that such learning systems are able to model simple dynamical systems such as pendulums, and can be combined with additional deep generative models to learn complex dynamics, such as video textures, in a fully end-to-end fashion.

In modern Reinforcement Learning, TD is combined with function approximation (i.e. neural networks) and off-policy learning. However, these three are known as the *deadly triad* [48, p. 264], because they may cause severe instability in the learning process Tsitsiklis and Van Roy [54]. While many variants of TD will provably converge despite the training instability, the quality of the solution at convergence is typically arbitrarily poor [24]. In the literature, there is a general belief that regularization can mitigate this instability, which is supported by basic analysis on the three standard examples.

However, this is not true! In the second chapter, we introduce a series of new counterexamples that are resistant to regularization. We demonstrate the existence of "vacuous" examples, which never do better than the limiting case regardless of the amount of regularization. This problem persists in most TD-based algorithms, which covers a wide swath of the RL literature; we make our analysis concrete by showing how this example forces the error bounds derived by Zhang, Yao, and Whiteson [63] to be extremely loose in practice. We further demonstrate that regularization is not monotonic in TD contexts, and that it is possible for regularization to increase error (or cause divergence) around some critical values. We extend these examples to the neural network case showing that these effects are not limited to the linear case and making the case for greater care in regularization in practical RL applications. Finally, there is a line of work starting with Emphatic-TD which seeks to stabilize off-policy training by resampling TD updates to appear on-policy. Contemporary Emphatic algortihms generally use a reversed version of TD to estimate the resampling function, which opens them up to instability from the same source as the original TD. We show that these techniques are similarly vulnerable. We show that regularization is not a panacea for stability in TD learning.

In the third chapter, we investigate new methods for stable TD learning that are resistant to off-policy divergence. Starting from an idea introduced by Kolter [24] we derive Projected Off-Policy TD, which reweighs TD updates to the closest distribution where the TD is non-expansive at the fixed point of its training. We learn the reweighing factors in the training loop using stochastic gradient descent (i.e. with time- and space-complexity comparable to learning the value function) and then apply those reweighing factors to each TD update. Crucially, this is distinct from contemporary work in the literature in that POP-TD does not resample to on-policy distribution, instead finding a "safe" distribution close to the data distribution.

Applying this to a novel offline RL example, we can clearly demonstrate how POP-TD mitigates the *distributional shift* between the dataset and the learned policy [30] while resampling as little as possible.

We conclude with a discussion on future directions that our work on stable models may take.

Chapter 1

Learning Provably Stable Deep Dynamics Models

Deep networks are commonly used to model dynamical systems, predicting how the state of a system will evolve over time (either autonomously or in response to control inputs). Despite the predictive power of these systems, it has been difficult to make formal claims about the basic properties of the learned systems. In this chapter, we propose an approach for learning dynamical systems that are guaranteed to be stable over the entire state space. The approach works by jointly learning an unconstrained dynamics model and Lyapunov function, then combining them in a novel reprojection layer to produce models that are guaranteed to be stable by construction everywhere in the state space, even without any training. We show that such learning systems are able to model dynamical systems such as compound pendulums and can be combined with additional deep generative models to learn ro generate images with complex dynamics such as video textures.

From "Learning Stable Deep Dynamics Models" by Manek and Kolter (2019)

1.1 Introduction

This chapter deals with the task of learning continuous-time dynamical systems. Given $x(t) \in \mathbb{R}^n$, a state at time t, we wish to model the time-derivative

$$\dot{x}(t) \equiv \frac{d}{dt}x(t) = f(x(t)) \tag{1.1}$$

for some function $f : \mathbb{R}^n \to \mathbb{R}^n$. Modeling the time evolution of such dynamical systems (or with control inputs $\dot{x}(t) = f(x(t), u(t))$ for $u(t) \in \mathbb{R}^m$ and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$) is a foundational problem in machine learning, with applications in reinforcement learning, control, forecasting, and many other settings. Owing to their representational power, neural networks have long been a natural choice for modeling the function f [14, 41, 37, 12]. However, when using a neural network to model dynamics in this setting very little can be guaranteed about the behavior of the learned system, especially about its stability. Informally, we say that a model is stable if we can pick a bounded set of states and guarantee that once the model enters that set it never leaves. While some recent work has begun to consider stability properties of neural networks [6, 45, 51], it has typically done so by softly enforcing stability as an additional loss term on the training data. Consequently, they can say little about the stability of the system in states outside the training data.

In this chapter, we propose an approach to learning neural network dynamics that are provably Lyapunov-stable over the entirety of the state space. We do this by jointly learning a nominal system dynamics and the certifying Lyapunov function, and then reprojecting the predictions of the nominal model onto the level set of the Lyapunov function. This stability is a hard constraint imposed upon the model: unlike recent approaches, we do not enforce stability via an imposed loss function but build it directly into the dynamics of the model. This means that even a randomly initialized model in our proposed model class will be provably stable everywhere in state space. The key to this is the design of a proper Lyapunov function, based on input convex neural networks [1], which ensures global exponential stability to an equilibrium point while still allowing for rich dynamics.

Using these methods, we demonstrate learning dynamics of physical models such as n-link pendulums, and show a substantial improvement over generic networks. We also show how such dynamics models can be integrated into larger network systems to learn dynamics over complex output spaces, combining the model with a variational auto-encoder (VAE) [23] to learn dynamic video textures [46].

1.2 Background and related work

Stability of dynamical systems. We consider the setting of uncontrolled[1] dynamics systems $\dot{x}(t) = f(x(t))$ for $x(t) \in \mathbb{R}^n$. Such a system is *globally asymptotically stable* around the equilibrium point $x_e = 0$ if we have $x(t) \to 0$ as $t \to \infty$ for any initial state $x(0) \in \mathbb{R}^n$; f is *locally asymptotically stable* if the same holds but only for $x(0) \in \mathcal{B}$ where $\mathcal{B}$ is some bounded set containing the origin. Similarly, f is *globally* or *locally* exponentially stable if trajectories approach to the origin is at some minimum rate:

$$\|x(t)\|_2 \leq m\|x(0)\|_2\, e^{-\alpha t} \tag{1.2}$$

for some constants $m, \alpha \geq 0$ for any $x(0) \in \mathbb{R}^n$ ($\mathcal{B}$, respectively).

The area of Lyapunov theory [20, 29] establishes the connection between these types of stability according to a *Lyapunov function*. Specifically, let $V : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable positive definite function, i.e., $V(x) > 0$ for $x \neq 0$ and $V(0) = 0$. Lyapunov analysis says that f is asymptotically stable, if and only if there exists some function V as above such the value of this function decreases along trajectories generated by f. Formally, this is the condition that the time derivative $\dot{V}(x(t)) < 0$, i.e.,

$$\dot{V}(x(t)) \equiv \frac{d}{dt}V(x(t)) = \nabla V(x)^T \frac{d}{dt}x(t) = \nabla V(x)^T f(x(t)) < 0 \tag{1.3}$$

This condition must hold for all $x(t) \in \mathbb{R}^n$ or for all $x(t) \in \mathcal{B}$ to ensure global or local

[1] We will discuss extending this to dynamics with control later; this is a non-trivial extension.

stability respectively. Similarly f is globally exponentially stable if and only if there exists positive definite V with a sufficiently steep gradient such that

$$\dot{V}(x(t)) \leq -\alpha V(x(t)), \quad \text{with } c_1\|x\|_2^2 \leq V(x) \leq c_2\|x\|_2^2. \tag{1.4}$$

Showing that these conditions imply the various forms of stability is relatively straightforward, but it is also true (but more complex to show) that any stable system must obey this property for some V. In this chapter our broad stategy is to construct a Lyapunov function and enforce conditions that ensure stability.

Stability of linear systems. For a linear system with matrix A

$$\dot{x}(t) = Ax(t) \tag{1.5}$$

it is well-known that the system is stable if and only if the real components of the the eigenvalues of A are all strictly negative. Equivalently, the same same property can be shown via a positive definite quadratic Lyapunov function

$$V(x) = x^T Q x \tag{1.6}$$

for $Q \in \mathbb{R}^{n \times n}, Q \succ 0$. In this case, by Equation 1.4, the following ensures global exponential stability:

$$\dot{V}(x(t)) = x(t)^T A^T Q x(t) + x(t)^T Q A x(t) \leq -\alpha x(t)^T Q x(t) \tag{1.7}$$

i.e., if we can find a positive definite matrix $Q \succeq I$ such that $A^T Q + QA + \alpha Q \preceq 0$ negative semidefinite. Such bounds (and much more complex extensions) for the basis for using linear matrix inequalities (LMIs), as a method to ensure stability of linear dynamical systems. The methods also have applicability to non-linear systems, and several authors have used LMI analysis to learn non-linear dynamical systems by constraining the linearized systems to have global Lyapunov functions [21, 2, 55], Even though the constraints

$$Q \succeq I, \quad A^T Q + QA + \alpha Q \preceq 0 \tag{1.8}$$

are convex in A and Q separately, they are not convex in A and Q jointly. Thus, the problem of jointly learning a stable linear dynamical system and its corresponding Lyapunov function, even for the simple linear-quadratic setting, is *not* a convex optimization problem, and alternative techniques such as alternating minimization need to be employed instead. Past work has considered different heuristics, such as approximately projecting a dynamics function A onto the (non-convex) stable set of matrices with eigenvalues $\mathsf{Re}(\lambda_i(A)) < 0$ [3]. It is no surprise, then, that learning stable non-linear systems is even more challenging:

Stability of non-linear systems For general non-linear systems, establishing stability via Lyapunov techniques is even more challenging. For the typical task here, which is that of establishing stability of some *known* dynamics $\dot{x}(t) = f(x(t))$, finding a suitable Lyapunov function is often more an art than a science. Although some general techniques such as sum-of-squares certification [43, 42] provide methods for certifying stability of polynomial (or similar) systems, these are often expensive and don't easily scale to high dimensional systems. Our proposed approach here is able to learn provably stable systems without solving this generally hard problem. While it is difficult to find a Lyapunov function that certifies the stability of some *known* system, we exploit the fact that it is relatively much easier to *enforce* some function to behave in a stable manner according to a Lyapunov function.

Lyapunov functions in deep learning Finally, there has been a small set of recent work exploring the intersection of deep learning and Lyapunov analysis [6, 45, 51]. Although related to our work here, the approach in this past work is quite different. As is more common in the control setting, these papers try to learn neural-network-based Lyapunov functions for control policies, but in way that enforces stability via a loss penalty. For instance Richards et al., [45] optimize a loss function that encourages $\dot{V}(x) \le 0$ for x in some training set. In contrast, our work guarantees stability everywhere in the state space, not just at a small set of points; but only for a simpler setting where the entire dynamics are to be learned (and hence can be 'constrained to be stable) rather than a stabilizing controller for known dynamics.

1.3 Joint learning of dynamics and Lyapunov functions

The intuition of the approach we propose in this paper is straightforward: instead of learning a dynamics function and attempting to separately verify its stability via a Lyapunov function, we propose to jointly learn a dynamics model and Lyapunov function, where the dynamics is inherently constrained to be stable (everywhere in the state space) according to the Lyapunov function.

Specifically, following the principles mentioned above, let $\hat{f} : \mathbb{R}^n \to \mathbb{R}^n$ denote a "nominal" unconstrained dynamics model, and let $V : \mathbb{R}^n \to \mathbb{R}$ be a positive definite function: $V(x) \ge 0$ for $x \ne 0$ and $V(0) = 0$. Then in order to (provably, globally) ensure that a dynamics function is stable, we can simply project $\hat{f}$ such that it points down the gradient of the Lyapunov function. This corresponds to the condition

$$\nabla V(x)^T \hat{f}(x) \le -\alpha V(x) \tag{1.9}$$

i.e., we define the dynamics

$$\begin{aligned} f(x) &= \mathsf{Proj}\left(\hat{f}(x), \{f : \nabla V(x)^T f \le -\alpha V(x)\}\right) \\ &= \begin{cases} \hat{f}(x) & \text{if } \nabla V(x)^T \hat{f}(x) \le -\alpha V(x) \\ \hat{f}(x) - \frac{\nabla V(x)}{\|\nabla V(x)\|_2^2}\left(\nabla V(x)^T \hat{f}(x) + \alpha V(x)\right) & \text{otherwise} \end{cases} \\ &= \hat{f}(x) - \frac{\nabla V(x)}{\|\nabla V(x)\|_2^2}\mathsf{ReLU}\big(\nabla V(x)^T \hat{f}(x) + \alpha V(x)\big) \end{aligned} \tag{1.10}$$

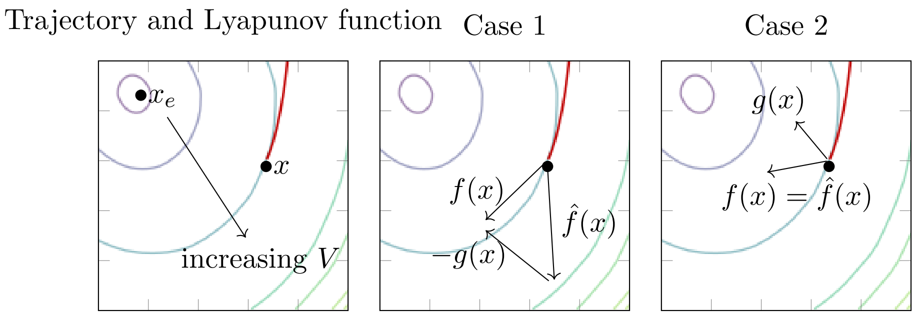


Figure 1.1: We plot the trajectory and the contour of a Lyapunov function of a stable dynamical system and illustrate our method. Let $g(x) = \frac{\nabla V(x)}{\|\nabla V(x)\|_2^2} \mathrm{ReLU}\left(\nabla V(x)^T \hat{f}(x) + \alpha V(x)\right)$. In the first case $\hat{f}(x)$ has a component $g(x)$ not in the half-space, which we subtract to obtain $f(x)$. In the second case $\hat{f}(x)$ is already in the half-space, so is returned unchanged.

where $\mathsf{Proj}(\mathsf{x}; \mathcal{C})$ denotes the orthogonal projection of x onto the point $\mathcal{C}$, and where the second equation follows from the analytical projection of a point onto a half-space. As long as V is defined using automatic differentiation tools, it is straightforward to include the gradient ∇V terms into the definition of f, and our final network can be trained just like any other function. The general approach here is illustrated in Figure 1.1.

1.3.1 Properties of the Lyapunov function V

Although the treatment above seems to make the problem of learning stable systems quite straightforward, the subtlety of the approach lies in the choice of the function V. Specifically, V needs to be positive definite and needs to have no local optima except the global optimum at 0. This is due to Lyapunov decrease condition: recall that we are attempting to guarantee stability to the equilibrium point $x = 0$, yet the decrease condition imposed upon the dynamics means that V is decreasing along trajectories of f. If V has a local optimum away from the origin, the dynamics may get stuck in this location; this manifests as the $\|\nabla V(x)\|_2^2$ term going to zero.

To enforce these conditions, we make the following design decisions regarding V:

No local optima. We represent V via an input-convex neural network (ICNN) function $g : \mathbb{R}^n \to \mathbb{R}$ [1], which enforces the condition that $g(x)$ be convex in its inputs x. Such a network is given by the recurrence

$$\begin{aligned} z_1 &= \sigma_0(W_0 x + b_0) \\ z_{i+1} &= \sigma_i(U_i z_i + W_i x + b_i) \qquad i \in \{1, \ldots, k-1\}. \\ g(x) &\equiv z_k \end{aligned} \tag{1.11}$$

For layer $i+1$: W_i are weights mapping from the input x to the $i+1$ layer activations; U_i are positive weights mapping previously layer activations z_i to the next layer; b_i are real-valued biases; and σ_i are convex, monotonically non-decreasing non-linear activations such as the ReLU or smooth variants. It is straightforward to show that with this formulation, g is convex in x [1], and indeed any convex function can be approximated by such networks [5].

Positive definite. The ICNN property enforces that V has only a single global optimum; for V to be positive definite, we must also enforce that this optimum is at $x = 0$. We could fix this by removing the bias term from Equation 1.11, but this would mean we could no longer represent arbitrary convex functions. We could also shift whatever global minimum to the origin, but that would require finding finding the global minimum during training, which itself is computationally expensive. Instead, we take an alternative approach: we shift the function such that $V(0) = 0$, and add a small quadratic regularization term to ensure strict positive definiteness.

$$V(x) = \sigma_{k+1}(g(x) - g(0)) + \epsilon \|x\|_2^2. \tag{1.12}$$

where σ_k is a positive convex non-decreasing function with $\sigma_k(0) = 0$, g is the ICNN defined previously, and ϵ is a small constant. These terms together still enforce (strong) convexity and positive definiteness of V.

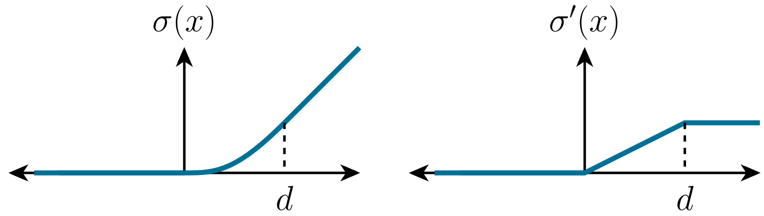


Figure 1.2: Rectified Huber Unit (ReHU), necessary for continuously differentiable Lyapunov functions.

Continuously differentiable. Although not always required, several of the conditions for Lyapunov stability are simplified if V is continuously differentiable. ReLU is discontinuous around 0, and the soft-plus smoothed ReLU is not zero at the origin. We use a smoothed version with quadratic knee in $[0, d]$, called the Rectified Huber Unit (ReHU):

$$\sigma(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x^2/2d & \text{if } 0 < x < d \\ x - d/2 & \text{otherwise} \end{cases}. \tag{1.13}$$

An illustration of this activation is shown in Figure 1.2.

Optionally warped input space. Our construction so far guarantees that the Lyapunov function has no local optima by making it convex. This is sufficient but not necessary, and it may even impose too strict a constraint on the learned dynamics. We can relax this function by allowing the input to the ICNN to be warped by any continuously differentiable invertible function $F : \mathbb{R}^n \times \mathbb{R}^n$. i.e., using

$$V(x) = \sigma_{k+1}(g(F(x)) - g(F(0))) + \epsilon\|x\|_2^2. \tag{1.14}$$

as the Lyapunov function. Invertibility ensures that the level sets of V, which are convex, map to contiguous regions of the composite function $g \circ F$. This allows the resultant Lyapunov function to be non-convex without having any optima other than the global.

With these conditions in place, we have the following result.

Theorem 1. *The dynamics defined by*

$$\dot{x} = f(x) \tag{1.15}$$

are globally exponentially stable to the equilibrium point $x = 0$. Where f is from Eqn. 1.10 and V is from Eqn. 1.12 or Eqn. 1.14, and $\hat{f}$ and V functions have finite, bounded weights.

Details. The proof is straightforward, and relies on the properties of the networks created above. First, note that by our definitions we have, for some M,

$$\epsilon\|x\|_2^2 \leq V(x) \leq M\|x\|_2^2 \tag{1.16}$$

where the lower bound follows from Eqn. 1.12 and the fact that g is positive. The upper bound follows from the fact that the ReHU activation is linear for large x and quadratic around 0. This in turn implies that $V(x)$ behaves linearly as $\|x\| \to \infty$, and is quadratic around the origin, so can be upper bounded by some quadratic $M\|x\|_2^2$.

The fact the V is continuously differentiable means that $\nabla V(x)$ (in f) is defined everywhere, bounds on $\|\nabla V(x)\|_2^2$ for all x follows from the the Lipschitz property of V, the fact that $0 \leq \sigma'(x) \leq 1$, and the $\epsilon\|x\|_2^2$ term

$$\epsilon\|x\|_2 \leq \|\nabla V(x)\|_2 \leq \sum_{i=1}^{k}\prod_{j=i}^{k}\|U_j\|_2\|W_i\|_2 \tag{1.17}$$

where $\|\cdot\|_2$ denotes the operator norm when applied to a matrix. This implies that the dynamics are defined and bounded everywhere owing to the choice of function $\hat{f}$.

Now, consider some initial state $x(0)$. The definition of f implies that

$$\frac{d}{dt}V(x(t)) = \nabla V(x)^T\frac{d}{dt}x(t) = \nabla V(x)^T f(x) \leq -\alpha V(x(t)). \tag{1.18}$$

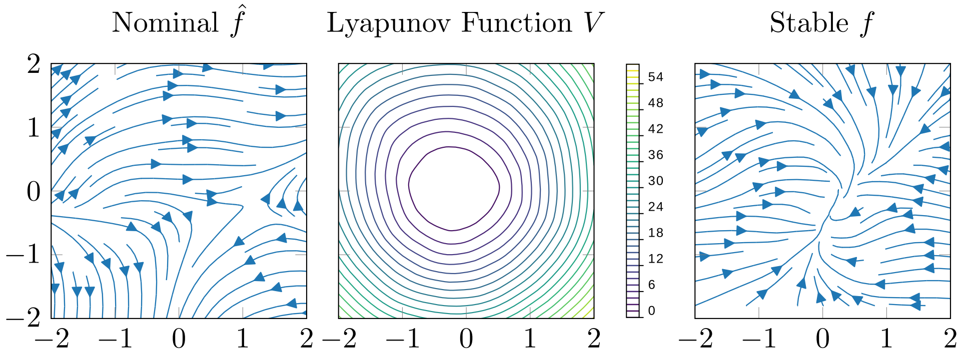


Figure 1.3: (left) Nominal dynamics $\hat{f}$ for random network; (center) Convex positive definite Lyapunov function generated by random ICNN with constraints from Section 1.3.1; (right) Resulting stable dynamics f.

Integrating this equation gives the bound

$$V(x(t)) \leq V(x(0))e^{-\alpha t} \tag{1.19}$$

and applying the lower and upper bounds gives

$$\epsilon \|x(t)\|_2^2 \leq M\|x(0)\|_2^2 e^{-\alpha t} \implies \|x(t)\|_2 \leq \frac{M}{\epsilon}\|x(0)\|_2 e^{-\alpha t/2} \tag{1.20}$$

as required for global exponential convergence. $\square$

1.4 Empirical results

We illustrate our technique on several example problems, first highlighting the inherent stability of the method for random networks, demonstrating learning on simple n-link pendulum dynamics, and finally learning high-dimensional stable latent space dynamics for dynamic video textures via a VAE model.

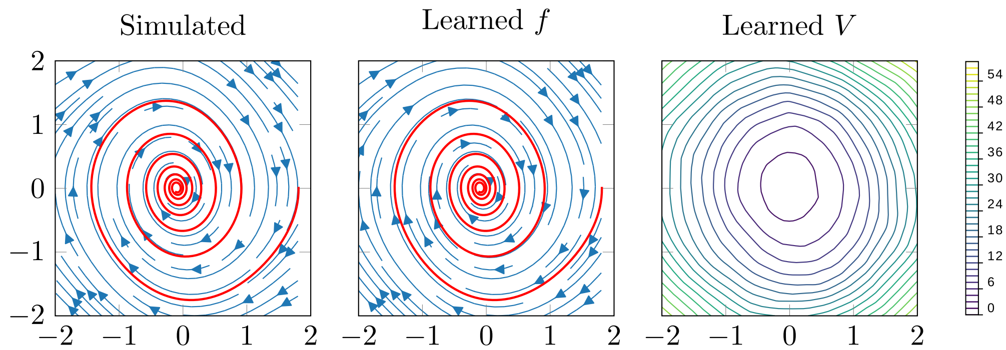


Figure 1.4: Dynamics of a simple damped pendulum. From left to right: the dynamics as simulated from first principles, the dynamics model f learned by our method, and the Lyapunov function V learned by our method (under which f is non-expansive).

1.4.1 Random networks

As a powerful visualization of the fact that our model is stable by construction, we can plot the dynamics created by random networks, i.e., without any training at all. Because the dynamics models are inherently stable, these random networks lead to stable dynamics with interesting behaviors, illustrated in Figure 1.3. Specifically, we let $\hat{f}$ be defined by a fully connected network and V be an ICNN. Both networks have two hidden layers with 100 nodes each, and are initialized by the Kaiming uniform initialization [18]. The U weights in the ICNN are further subject to a softplus unit to make them positive.

1.4.2 n-link pendulum

Next we look at the ability of our approach to model a dynamical system from kinematics, specifically the n-link pendulum. A damped, rigid n-link pendulum's state x can be described by the angular position θ_i and angular velocity θ_i of each link i. As before $\hat{f}$ and the Lyapunov function V have two hidden layers of 100 nodes, with properties described in Section 1.3.1. Models are trained with pairs of data $(x, \dot{x})$ produced by the symbolic algebra solver `sympy`, using simulation code adapted from [56].

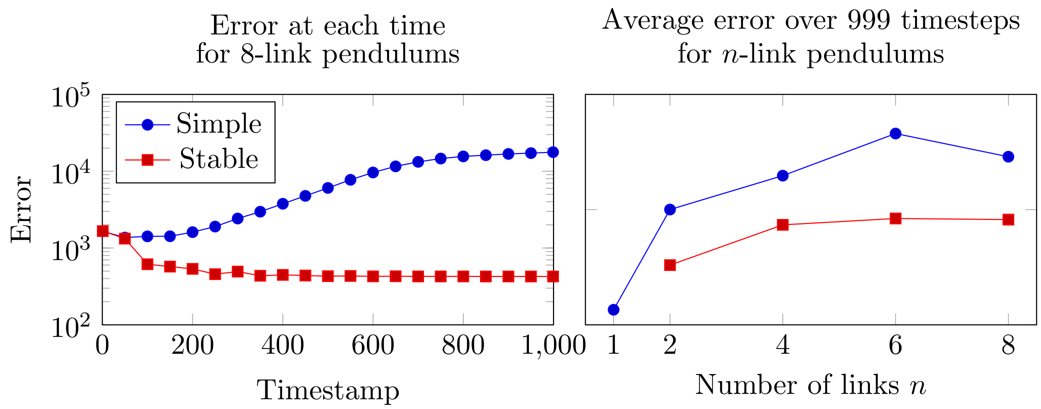


Figure 1.5: Error in predicting $\theta, \dot{\theta}$ in 8-link pendulum at each timestep (left); and average error over 999 timesteps as the number of links in the pendulum increases (right).

In Figure 1.4, we compare the simulated dynamics with the learned dynamics in the case of a simple damped pendulum (i.e. with $n = 1$), showing both the vector field and a single simulated trajectory, and draw a contour plot of the learned Lyapunov function. As seen, the system is able to learn dynamics that can accurately predict motion of the system even over long time periods. We can also recover the laws of conservation of energy implicit in the data, including the fact that kinetic energy is lost slowly but not potential energy.

We also evaluate the learned dynamics quantitatively varying n and the time horizon of simulation. Figure 1.5 presents the total error over time for the 8-link pendulum, and the average cumulative error over 1000 time steps for different values of n. While both the simple and stable models show increasing mean error at the start of the trajectory, our model is able to capture the loss of energy in the physical system and in fact exhibits decreasing error towards the end of the simulation. In comparison, the error in the simple model increases.

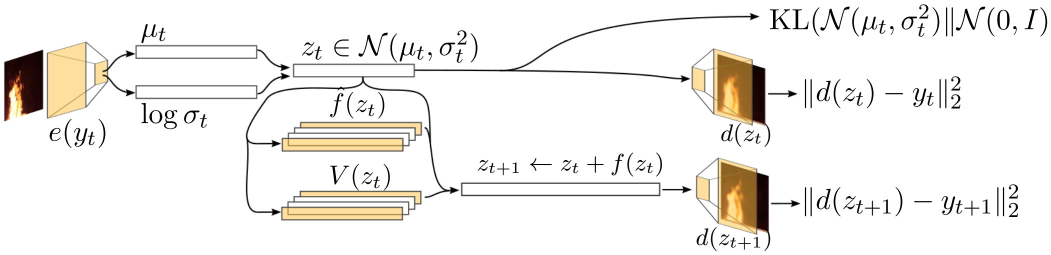


Figure 1.6: Structure of our video texture generation network. The encoder e and decoder d form a Variational Autoencoder, and the stable dynamics model f is trained together with the decoder to predict the next frame in the video texture.

1.4.3 Video Texture Generation

Finally, we apply our technique to stable video texture generation, using a Variational Auto-Encoder (VAE) [23] to learn an encoding for images, and our stable network to learn a dynamics model in the latent space. Given a sequence of frames $(y_0, y_1, \ldots)$, we feed the network the frame at current time t and train it to reconstruct the frames at the current time t and subsequent time-step $t+1$. Specifically, we consider a VAE defined by the encoder $e : \mathcal{Y} \to \mathbb{R}^{2n}$ giving mean and variance $\mu, \log \sigma_t^2 = e(y_t)$, latent state $z_t \in \mathbb{R}^n \sim \mathcal{N}(\mu_t, \sigma_t^2)$, and decoder $d : \mathbb{R}^n \to \mathcal{Y}$, $y_t \approx d(z_t)$. We train the network to minimize both the standard VAE loss (reconstruction error plus a KL divergence term), but *also* minimize the reconstruction loss of a next predicted state. We model the evolution of the latent dynamics at $z_{t+1} \approx f(z_t)$, or more precisely $y_{t+1} \approx d(f(z_t))$. In other words, as illustrated in Figure 1.6, we train the full system to minimize

$$\underset{e,d,\hat{f},V}{\text{minimize}} \sum_{t=1}^{T-1} \left(\mathsf{KL}(\mathcal{N}(\mu_t, \sigma_t^2 I \| \mathcal{N}(0, I)) + \mathbf{E}_z \left[\|d(z_t) - y_t\|_2^2 + \|d(f(z_t)) - y_{t+1}\|_2^2 \right] \right) \tag{1.21}$$

We train the model on pairs of successive frames sampled from videos. To generate video textures, we seed the dynamics model with the encoding of a single frame

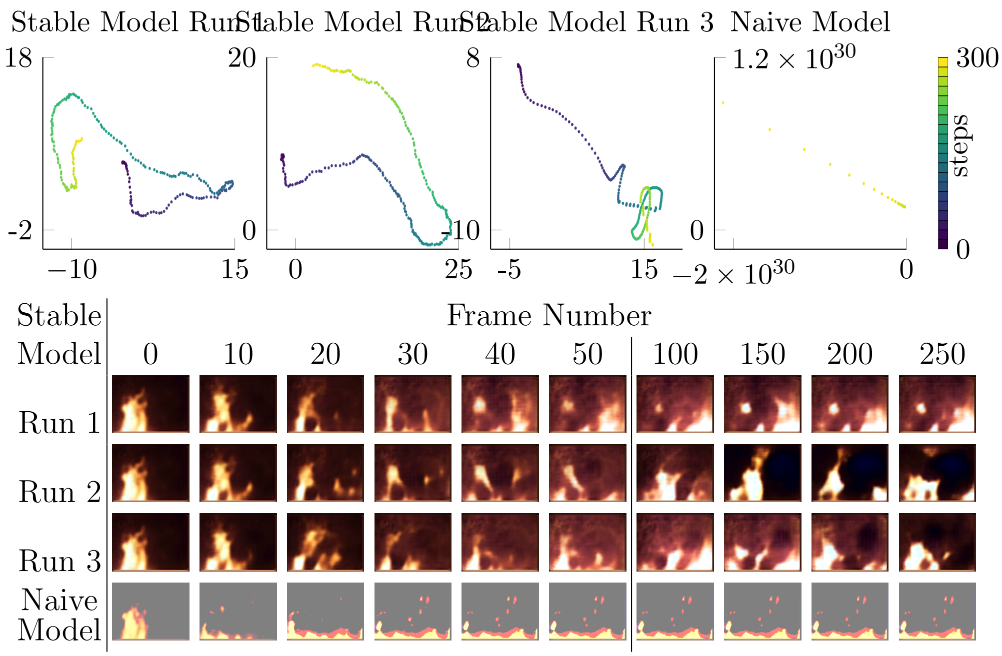


Figure 1.7: Samples generated by our stable video texture networks, with associated trajectories above. The true latent space is 320-dimensional; we project the trajectories onto a two-dimensional plane for display. For comparison, we present the video texture generated using an unconstrained neural network in place of our stable dynamics model.

and numerically integrate the dynamics model to obtain a trajectory. The VAE decoder converts each step of the trajectory into a frame. In Figure 1.7, we present sample stable trajectories and frames produced by our network. For comparison, we also include an example trajectory and resulting frames when the dynamics are modelled without the stability constraint (i.e. letting f in the above loss be a generic neural network). For the naive model, the dynamics quickly diverge and produce a static image, whereas for our approach, we are able to generate different (stable) trajectories that keep generating realistic images over long time horizons. We control the "temperature" of the generation process by adding controlled amounts of random noise to the system at each step.

1.5 Conclusion

We proposed a method for learning provably stable non-linear dynamical systems using neural networks. The approach jointly learns a convex positive definite Lyapunov function along with dynamics constrained to be stable according to these dynamics everywhere in the state space. We show that these models can be integrated into other deep architectures such as VAEs, and learn complex latent space dynamics is a fully end-to-end manner. Although we have focused here on the autonomous (i.e. uncontrolled) setting, the method opens several directions for future work, such as integration into dynamical systems for control or for model-based reinforcement learning settings. Having stable dynamics as a neural-network primitive can be useful in many diverse contexts, and combining these stable systems with the representational power of deep networks offers a powerful tool in modeling dynamical systems.

1.6 Adaptation to Stable Control and RL

After the successes of our stable dynamics model, we attempted to extend it to also learn stable policies and value functions. The intuitive extension to this is to replace the dynamics model $\hat{f}$ with fixed (known) dynamics $\tilde{f}$ and a learnable policy network

π. That is, we train to minimize:

$$\mathsf{ReLU}\big(\nabla V(x)^T \tilde{f}(x,\pi) + \alpha V(x)\big) \tag{1.22}$$

given traces from simulated dynamics. We also transformed the dynamics so that the goal state was positioned at the origin, choosing suitable transformations for the dynamics and Lyapunov functions. As required by the approach, we attempted to train it from trajectory samples to minimize the error over one step.

We were able to successfully learn stabilizing controllers for toy examples such as a simple damped pendulum and for the cartpole problem. Unfortunately, we were not able to learn a swing-up controller for either environment, or any type of controller for an Acrobot[2] or more complex locomotion tasks. We observed that the training would consistently fail in the same way: the nominal dynamics function would diverge to the point of uselessness, followed by the learned Lyapunov function collapsing to a trivial function.

This persists despite any amount of regularization, hyperparameter tuning, and even across a variety of environments. Contemporary efforts in the literature were similarly unable to scale this approach to locomotion tasks. The consistent failure of this method suggested that an underlying principle was being violated, and that regularization was not able to address that. We eventually investigated how the difference in distributions between the data used to train the purportedly stable controller and the policy the controller was attempting to learn, which led us to the work in the next chapter.

[2] A two-link pendulum with a single actuator in the middle joint. A pendulum with an actuator at the fixed joint is a Pendubot [52].

Chapter 2

The Pitfalls of Regularization in Off-Policy TD Learning

Temporal Difference (TD) learning is ubiquitous in reinforcement learning, where it is often combined with off-policy sampling and function approximation. Unfortunately, this combination of conditions (the *deadly triad*) often leads to unstable training and unbounded error. Modern RL methods often implicitly assume that regularization is sufficient to mitigate the problem and the standard deadly triad examples from the literature are not able to refute this. In this chapter, we introduce a series of new counterexamples to show that this problem is not solved by regularization. We show that TD methods can fail to learn a non-trivial value function under *any* amount of regularization, that regularization can itself induce divergence; and we show that one of the most promising mitigations (Emphatic-TD algorithms) may also diverge under regularization. We further demonstrate such divergence when using neural networks as function approximators. Thus, we argue that there needs to be much more care in the application of regularization to RL methods.

From "The Pitfalls of Regularization in Off-Policy TD Learning" by Manek and Kolter (2022)

2.1 Introduction

Temporal Difference (TD) learning is a method for learning expected future-discounted quantities from Markov processes, using transition samples to iteratively improve estimates. This is most commonly used to estimate expected future-discounted rewards (the *value function*) in Reinforcement Learning (RL). Advances in RL allow us to use powerful function approximators, and also to use "off-policy" sampling strategies (i.e. which do not naively follow the underlying Markov process.) When TD, function approximation, and off-policy training are all combined, learned functions exhibit severe instability and divergence, as classically observed by Williams and Baird III [59] and Tsitsiklis and Van Roy [54]. This combination is known in the literature as the *deadly triad* [48, pg. 264], and while many contemporary variants of TD are designed to converge despite the instability, the quality of the solution at convergence may be arbitrarily poor [24].

A common technique to avoid unbounded error is ℓ_2 *regularization* [53], i.e. penalizing the squared norm of the weights in addition to the TD error. This is generally understood to bound the worst-case error in exchange for biasing the model and potentially increasing the error everywhere else. When used on three common examples of the deadly triad [24, 59, 48, pg. 260], regularization appears to mitigate the worst aspects of the divergence in practice. Consequently, it has become an essential assumption made by many RL algorithms [8, 33, 50, 61, 64, 63, 27] and is seen as routine and innocuous.

We argue that this perspective on regularization in off-policy TD is fundamentally mistaken. While regularization is indeed well-behaved and innocuous in classic fully-supervised contexts, the use of bootstrapping in TD means that even small amounts of model bias induced by regularization can cause divergence. This is an oft-ignored phenomenon in the literature, and so we introduce a series of new counterexamples (summarized in Table 2.1) to show how regularization can have counterintuitive and destructive effects in TD. We show that vacuous solutions and training instability are *not* solved by the use of regularization; that applying regularization can sometimes induce divergence and increase worst-case error; and that Emphatic-TD based algorithms—which are the most promising way to correct stability from off-policy training—can themselves diverge when regularized. We finally also illustrate misbehaving regularization in the context of neural network value function approximation, demonstrating the general pitfalls of regularization possible in RL algorithms. Regularization needs to be treated cautiously in the context of RL, as it behaves differently than in supervised settings.

Our counterexamples demonstrate these core ideas:

TD learning off-policy can be unstable and/or have unbounded error even when it converges. Following well-established methods we show there is some off-policy distribution under which TD with linear value function approximation diverges *and* learns a model with unbounded error (even if it were able to converge to the TD fixed point). This concisely demonstrates key features of the training error: the error is small when the distribution is close to on-policy, but the error diverges around specific off-policy distributions. The intuition behind this, explained in Section 2.3, is that the off-policy[1] TD update involves a projection operation that depends on the sampling distribution and can be arbitrarily far away from the true value. This basic fact has already been established by past work [59, 24], but our example is based upon a particular simple three-state MP, drawn in Figure 2.1a.

Regularization cannot always mitigate off-policy training error. We next introduce regularization into our setting, and show how it changes the relationship between training error and off-policy training. As explained in Section 2.2, we penalize the ℓ_2-norm of learned (linear) weights with some coefficient η; as η increases, the learned weights approach zero. However, in **Example 1**, we show that there exists an off-policy distribution such that for any non-negative η, the regularized TD fixed point attains strictly higher approximation error than the zero solution (i.e., the infinitely regularized point). We call such examples *vacuous*. In other words, vacuous

[1]We consider a sampling distribution to be *on-policy* if it follows the stationary distribution of the MP and *off-policy* otherwise; we do not explicitly consider a separate policy in this chapter.

value functions never do better than guessing zero for all states, for any amount of regularization.

We further analyze this vacuous example in the context of the algorithm in [63]. In this work, the authors assume the use of regularization to derive bounds on the learned error under off-policy sampling. Although these bounds are technically correct in the case of our counterexample, they are very loose, at about 2000 times the limit of vacuity. This highlights the challenge of formally relying on regularization to bound model error, and illustrates the danger of relying on regularization in theoretical RL.

Small amounts of regularization can cause model divergence. There is a general implicit assumption in much ML literature that regularization monotonically shrinks learned weights and consequently the model output. This intuition comes from classic fully-supervised machine learning where it typically holds. But because TD bootstraps value estimates (i.e. learns values using its own output), the regularizer is composed arbitrarily often, and so it is possible for small amounts of bias to be arbitrarily magnified. We dub this phenomenon "small-eta divergence" and illustrate it in **Example 2**. We relate this to the presence of negative eigenvalues in an intermediate step of the solution and show that, in some settings, the error of the TD solution may be relatively small when applied with no regularization but adding regularization causes the model to have *worse* error than the zero solution.

One common solution to this problem is to lower-bound η to guarantee that regularization behaves monotonically. However, we further show that such a lower bound may restrict the model to a domain in which the model is vacuous. That is, a model that is not vacuous becomes vacuous when regularized with this lower bound. We also show that it is not always possible to select a single η *a priori*, with examples of mutually-incompatible off-policy distributions where there is no η that achieves better than vacuous or nearly-vacuous results at different distributions.

Emphatic-TD-based algorithms are vulnerable to instability from regularization. Emphatic-TD [49] attempts to solve the problem of training off-policy by resampling TD updates so they appear to be on-policy. This technique requires an emphasis model that decides how to scale each TD update, and learning this has been the key challenge preventing widespread adoption of Emphatic-TD. A recent paper [64] proposed learning this emphasis model using "reversed" TD while simultaneously learning the value model using regular TD. The resultant algorithm is called COF-PAC, and employs regularization to ensure that the two TD models eventually converge.

We show that regularization, while necessary, can be harmful for such models in **Example 3**. Specifically, we construct a model that converges to the correct solution without regularization but to an arbitrarily poor solution when regularized. The intuition behind this is that regularizing the emphasis model changes the effective distribution of the TD updates to the value model, which can cause the value model to have arbitrarily large error. We complete the example by showing that regularizing the value function separately does not restore performance.

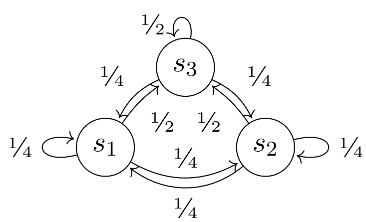
Regularization can cause model divergence in neural networks. So far most analysis of the deadly triad in the literature focuses on the linear case. We extend our example to a nine-state Markov chain (shown in Figure 2.8), and show how the previously identified problems persist into the neural network case in **Example 4**. We show two key similarities: first, models trained at certain off-policy distributions may be vacuous. Second, small amounts of regularization counterintuitively *increase* error. This illustrates Example 2 in the NN case.

2.2 Preliminaries and Notation

Consider the n-state Markov chain $(\mathcal{S}, P, R, \gamma)$, with state space $\mathcal{S}$, state-dependent reward $R : \mathcal{S} \to \mathbb{R}$, and discount factor $\gamma \in [0, 1]$. $P \in \mathbb{R}^{n \times n}$ is the transition matrix, with P_{ij} encoding the probability of moving from state i to j. We wish to estimate the value function $V : \mathcal{S} \to R$, defined as the expected discounted future reward of being in each state: $V(s) \doteq \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \middle| s_0 = s\right]$. A key property is that it follows

Example 1	There exist off-policy distributions under which TD learns a *vacuous* model (one which—despite any amount of regularization—never does better than guessing zeros).
Example 2	Small values of the regularization parameter η can make TD diverge in models that otherwise converge. This is an unavoidable effect of bootstrapping in TD, and setting a lower-bound to exclude this may render models vacuous.
Example 3	Emphatic-TD-inspired algorithms are a promising way to reweigh samples and mitigate the effects of training off-policy. However, if this reweighing is learned using TD, then regularization can bias the emphasis model and cause the value model itself to diverge.
Example 4	Training instability and increased error due to the deadly triad also occur when neural networks are used. We construct an empirical example and draw qualitative comparisons.

Table 2.1: Summary of theorems.



(a) Three-state MP.

$$\frac{1}{4}\begin{bmatrix}1 & 1 & 2\\ 1 & 1 & 2\\ 1 & 1 & 2\end{bmatrix} \tag{2.1}$$

(b) Three-state MP.

Figure 2.1: Our three-state counterexample Markov Process. We use this to illustrate how TD models can fail despite common mitigating strategies with linear function approximation.

the Bellman equation:

$$V = R + \gamma PV \tag{2.2}$$

Using linear function approximation to learn V, we assume a matrix of feature-vectors $\Phi \in \mathbb{R}^{n\times k}$ that is fixed, and a vector of parameters $w \in \mathbb{R}^k$ that is learned. The Bellman equation is then:

$$\Phi w = R + \gamma\, P\, \Phi w \tag{2.3}$$

When w is learned with TD, this equation is only valid if the TD updates are *on-policy* (that is, they are distributed according to the steady-state probability of visiting each state, written as $\pi \in \mathbb{R}^n$). In the general case, where TD updates follow a (possibly) different distribution $\mu \in \mathbb{R}^n_0$, the TD solution is a fixed point of the Bellman operator followed by a projection [24]:

$$\Phi w = \Pi_\mu \left(R + \gamma P \Phi w\right) \tag{2.4}$$

where the matrix $\Pi_\mu = \Phi(\Phi^\top D\Phi)^{-1}\Phi^\top D$ projects the Bellman backup onto the column-space of Φ, reweighed by the state-distribution matrix $D = \mathrm{diag}(\mu)$. This yields the closed-form solution:

$$w = A^{-1}\vec{b} \tag{2.5}$$

Where $A = \Phi^\top D(I - \gamma P)\Phi$ and $\vec{b} = \Phi^\top DR$. When this solution is subject to ℓ_2 regularization, some non-negative η is added to ensure the matrix being inverted is positive definite:

$$w^*(\eta) = (A + \eta I)^{-1}\vec{b} \tag{2.6}$$

As will be important later, we note that as η increases it drives $w^*(\eta)$ towards zero.

2.3 Our Counterexamples

Under deadly triad conditions are present, TD may learn a value function with arbitrarily large error even if the true value function can be represented with low error. Consider the three-state MP in Figure 2.1a, which we instantiate with the value function $V = [1,\ 2.2,\ 1.05]^\top$ and discount factor $\gamma = 0.99$. The reward function is computed as $R \leftarrow (I - \gamma P)V$. We choose a basis Φ with small representation error $\|\Pi_\mu V - V\| \leq \epsilon$:

$$\Phi = \begin{bmatrix} 1 & 0 \\ 0 & -2.2 \\ {}^1\!/\!{}_2(1.05+\epsilon) & -{}^1\!/\!{}_2(1.05+\epsilon) \end{bmatrix} \qquad \text{where } \epsilon > 0 \tag{2.7}$$

We first consider the unregularized ($\eta = 0$) case, closely following the derivation in [24]. We wish to show there is some sampling distribution μ such that error in the learned value function is unbounded. To do this, we set $\mu = [0.56(1-p), 0.56p, 0.44]$, where $p \in (0, 1)$. We set $\epsilon = 10^{-4}$ and find p around which A is ill-conditioned by solving $\det(A) = 0$:

$$p = 0.102631 \qquad \vee \qquad p = 0.807255 \tag{2.8}$$

A^{-1} (and consequently the TD error) can be made arbitrarily large by selecting p close to these values, which completes the introductory example. Now we look at the behavior of TD under regularization, which is the main contribution of this chapter.

2.3.1 Regularization cannot always mitigate the error from training off-policy.

There is a belief in the literature that regularization is a trade-off between reducing the blow-up of asymptotic errors and accurately learning the value function everywhere else [8, 63]. However, this belief does not accurately capture the nature of regularization: we show that it is possible to learn models that never perform better than always guessing zero despite any amount of regularization. That is, the TD error at all η is at least as much as the error as $\eta \to \infty$. We call such models *vacuous*.

Example 1. We use the same setting as in Section 2.3. When TD is regularized, there may exist some off-policy distribution at which TD learns a vacuous model. In notation:

$$\|\Phi w^*(\eta) - V\| \geq \lim_{\eta\to\infty} \|\Phi w^*(\eta) - V\| = \|\Phi\vec{0} - V\| = \|V\| \qquad \forall \eta \in \mathbb{R}_0^+ \tag{2.9}$$

Details. We use the same setting as in Section 2.3. We observe that $\hat{w} = [1, -1]^\top$ minimizes the least-squares error $\|\Phi\hat{w} - V\|$, and further observe that a sufficient condition for a solution to be vacuous is that $\hat{w}^\top w^*(\eta) \leq 0$. Solving:

$$0 = \hat{w}^\top w^*(\eta) = \frac{\eta p - 0.233\eta - 0.304p^2 + 0.276p - 0.025}{\eta^2 + 1.44\eta p + 0.215\eta - 0.193p^2 + 0.175p - 0.016} \tag{2.10}$$

$$\implies p \in \{0.102636, \dots\} \tag{2.11}$$

We verify that TD is vacuous at $p = 0.102636$ by computing the TD error at convergence:

$$\|\Phi w^*(\eta) - V\|^2\Big|_{p=\tilde{p}} = \frac{\eta^2(0.148 + 0.744\eta + \eta^2)}{\eta^2(0.132 + 0.727\eta + \eta^2)}\|V\|^2 \geq \|V\|^2 \quad (\forall \eta \in \mathbb{R}^+) \tag{2.12}$$

Since the fraction term in Equation 2.12 is obviously improper, we can conclude that our example will always have at least $\|V\|$ error over all η, and is therefore vacuous. □

We note that the error is not defined at $\eta = 0$ because this corresponds to a model divergence similar to our introductory example. In practice, the TD fixed point will still converge to a vacuous solution:

$$\lim_{\eta\to 0} \|\Phi w^*(\eta) - V\|^2 = \frac{0.148}{0.132}\|V\|^2 > \|V\|^2 \tag{2.13}$$

Geometry of vacuous linear models.

We begin by noting that we can easily find the solution $\hat{w}$ that minimizes the least-squares error $\|\Phi\hat{w} - V\|$. If we consider this solution as a vector (as drawn in Figure 2.2a), we can immediately see that there is an ℓ_2-ball around $\hat{w}$ corresponding to the set of $w^*(\eta)$ with no more than $\|V\|$ error.

Similarly, we can trace the trajectory that the TD solution $w^*(\eta)$ takes as η is increased from 0 to ∞. We know that, as $\eta \to \infty$, $w^*(\eta)$ is crushed to zero and so all trajectories must eventually terminate at the origin. When regularized models are not vacuous, the trajectory intersects the non-vacuous-error ball. We see this in trajectory 2, where the error briefly dips below $\|V\|$ in Figure 2.2b.

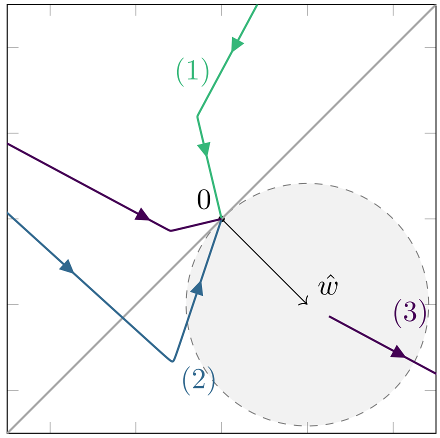
Intuitively, a sufficient condition for a solution to be vacuous is that it remains in the half-space that is tangent to and excludes the non-vacuous parameter ball. This is equivalent to finding some distribution μ such that $\hat{w}^\top w^*(\eta) \leq 0$ for all η, which we numerically solve to obtain the model in trajectory 1. From Figure 2.2a we can see the trajectory remains in the half-space, and from Figure 2.2b we can see that the error is never less than $\|V\|$. Trajectory 1 is a vacuous example.

We observe that Example 1, because it remains entirely in the half-space $\hat{w}^\top w^*(\eta) \leq 0$, could easily be generalized to any form of convex regularization such as $\ell_1, \ell_2, \ell_\infty$, etc. We leave this for future work.

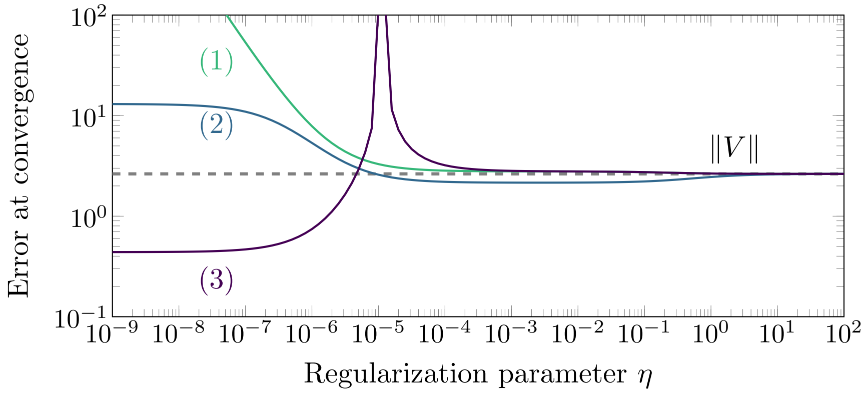
This intuition does not persist in the neural network case (discussed in Section 2.3.4). In that case, the relationship between parameters and error does not admit a clean non-vacuous ball, but instead a deeply non-linear set of states. The resultant geometry does not admit a clean, intuitive, explanation.

A second example.

We present a second example where the error is stationary with respect to the regularization parameter. This is worse than Example 1 because we are able to show that the point the model converges to is *independent* of regularization. This example



(a) As η increases, $w^*(\eta)$ traces different trajectories at different μ. $\hat{w}$ minimizes the error, and we shade the area with TD error less than $\|V\|$.



(b) We plot the error curves corresponding to the three $w^*(\eta)$ trajectories, along with $\|V\|$. Trajectory 1 is vacuous because the error is at least $\|V\|$ for all η.

Figure 2.2: Plotting the trajectory of the parameters on above and the errors below, we show how our counterexample 1 is never better than $\|V\|$ because it remains in half-space where $\hat{w}^\top w^*(\eta) \leq 0$. For comparison, we show trajectory 2 that is improved by regularization, and 3, which exhibits small-η errors. (The trajectories are distorted, so the errors in the two plots are not directly comparable.)

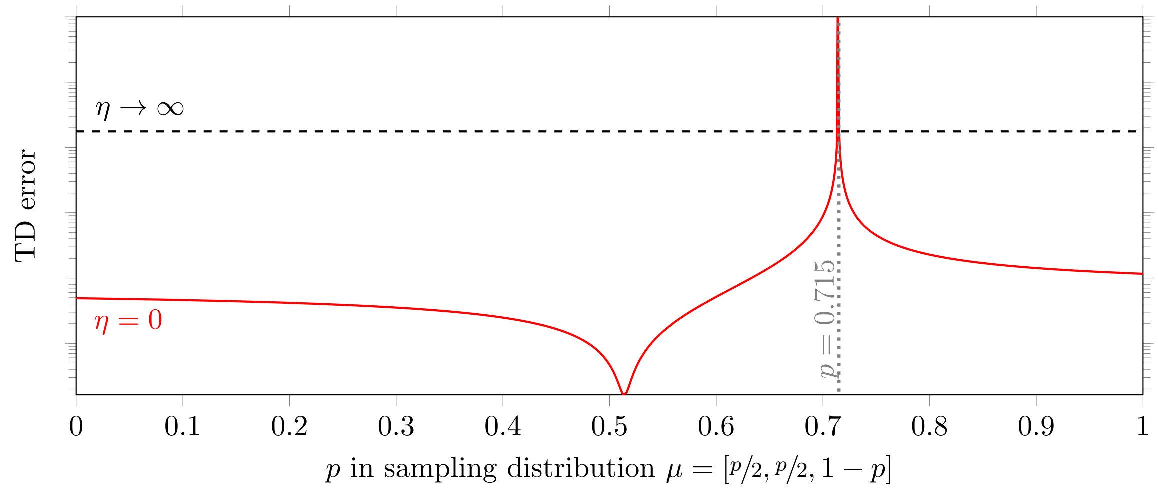


Figure 2.3: We plot TD error against p for our three-state MP with $\epsilon = 10^{-4}$. This shape is similar to that in [24]. There is a minima close to π ($p \approx 0.5$), and an asymptote at the singularity ($p \approx 0.715$). At different levels of regularization the error function moves between the unregularized case ($\eta = 0$) and the limiting case ($\eta \to \infty$), as analyzed in Section 2.3.1. We show that there is some p at which the error is never below the $\eta \to \infty$ line.

is the natural three-state extension to the two-state counterexample by Kolter [24].

Details. We use the same setting as in Section 2.3, except the value function is $V = [1,\ 1,\ 1.05]^\top$ and basis Φ selected to have small representation error $\|\Pi_D V - V\| \le \epsilon$:

$$\Phi = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ {}^1\!/\!{}_2(1.05+\epsilon) & -{}^1\!/\!{}_2(1.05+\epsilon) \end{bmatrix} \qquad \text{where } \epsilon > 0 \tag{2.14}$$

We set $\epsilon = 10^{-4}$ and write down $w^*(\eta)$ in terms of g, a scalar function of η and p:

$$w^*(\eta) = (A + \eta I)^{-1}\vec{b} = \frac{(2\eta + p)(0.925 - 1.29p)}{100\eta^2 + 47.4p\eta + 1.85\eta - 1.30p^2 + 0.927p} \cdot \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{2.15}$$

$$\equiv g(p,\eta) \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{2.16}$$

When $g(p,\eta) \le 0$, the TD solution is vacuous. We show that directly:

$$\|\Phi w^*(\eta) - V\| = \|g(p,\eta)\Phi * [1,-1]^\top - \Phi * [1,-1]^\top\| = \|g(\eta) - 1\| \cdot \|V\| \tag{2.17}$$

When $g(p,\eta) \le 0$, then $\|g(p,\eta) - 1\| \ge 1$ for all η and the TD solution is vacuous. We find such a solution by noting the numerator has two roots in p, one of which corresponds to a vacuous solution: $g(0.715083, \eta) = 0$ $(\forall \eta)$, and this completes the example!

In this setting, when TD updates follow the sampling distribution $p \approx 0.715083$, the error of the model at convergence is always $\|V\|$ regardless of regularization. Our example converges to the same vacuous value regardless η. $\square$

We present this graphically in Figure 2.3, where we plot the relationship between the off-policy distribution and the error at the TD fixed point. We plot the error with no regularization ($\eta = 0$) and the limiting error ($\eta \to \infty$).

We can see that the TD error intersects the $\eta \to \infty$ line immediately before and after the singularity. Our counterexample corresponds to the second root (that is, the intersection point at higher p.) That corresponds to the stationary point between the asymptote that is crushed and the error on the right that increases. If our simpler derivation proved unsatisfying, we can also derive this counterexample using this fact:

$$0 = \frac{d}{d\eta}\hat{w}^\top w^*(\eta) = \frac{p(p-0.715083)}{p(p-0.714303)^2} \tag{2.18}$$

From this, we can easily see that the counterexample is at $p = 0.715083$. And this completes the example! We have discovered some p at which the TD error is always at least $\|V\|$, regardless of regularization, and so our example learns a vacuous value function.

Breaking the Deadly Triad **and our counterexample.**

In light of our example we examine the work of [63] in which the authors derive a bound for the regularized TD error under a novel double-projection update rule. We apply our example to their bound and show that their method may produce loose bounds on TD solutions, and so doesn't quite break the deadly triad:

$$\|\Phi w^*(\eta) - V\| \leq \frac{1}{\xi}\left(\frac{\sigma_{\max}(\Phi)^2}{\sigma_{\min}(\Phi)^4\sigma_{\min}(D)^{2.5}} \cdot \|V\|\eta + \|\Pi_D V - V\|\right) \tag{2.19}$$

for $\xi \in [0,1]$, where $\sigma_{\max}$ and $\sigma_{\min}$ denote the largest and smallest singular value respectively. Theorem 2 from [63] bounds η, and therefore also b:

$$\eta > \arg\inf_\eta \|\Phi - C_0\| = {}^{0.177}/_{(1-\xi)^2} \tag{2.20}$$

$$\inf_\xi b(\xi,\eta) = 5.20 \times 10^4 \approx 2000 * \|V\| \tag{2.21}$$

Their method bounds the error in our example by $2000 * \|V\|$, which is tremendously loose.

Analyzing the second example starting from Equation 2.19:

$$\|\Phi w^*(\eta) - V\| \leq b(\eta, \xi) = \frac{1}{\xi}\left(\frac{\sigma_{\max}(\Phi)^2}{\sigma_{\min}(\Phi)^4 \sigma_{\min}(D)^{2.5}} \cdot \|V\|\eta + \|\Pi_D V - V\|\right) \tag{2.22}$$

$$= {}^1\!/\!_\xi \cdot (38.0\eta + 8.07 \times 10^{-5}) \tag{2.23}$$

for $\xi \in [0, 1]$, where $\sigma_{\max}$ and $\sigma_{\min}$ denote the largest and smallest singular value respectively. Theorem 2 from [63] bounds η, and therefore also b:

$$\eta > \arg\inf_{\eta} \|\Phi - C_0\| = 0.367(6.86 - 13.7\xi + 6.86\xi^2)^{-1} \tag{2.24}$$

$$\inf_{\xi} b(\xi, \eta) = 13.8 = 7.86 * \|V\| \tag{2.25}$$

Under our example, their method bounds the error at no more than $7.86 * \|V\|$, which is a very loose bound that permits vacuous solutions. This illustrates the risk of trying to regularize away singularities, particularly in theoretical work.

Investigating the cause of the loose bounds reveals that the presence of $\sigma_{\min}(D)^{2.5}$ in 2.19 is largely responsible. As D is a diagonal matrix encoding the sampling distribution, $\sigma_{\min}(D)$ is the smallest sampling rate of any state, and so the bound must be at least $\frac{\eta}{\xi n^{2.5}}$ for any perfectly representable n-state MP. Unfortunately, this appears to be fundamental limit caused by finding a linear bound to an error that scales non-linearly, and following their derivation does not readily admit a way to improve this.

2.3.2 Small amounts of regularization can cause large increases in training error.

There is a general assumption in the literature that ℓ_2 regularization monotonically shrinks the learned weights. While this is true in classification, regression, and other non-bootstrapping contexts, this is not true in TD. Because TD bootstraps values, it composes the regularization over itself arbitrarily deep, and so model bias may be arbitrarily magnified.

This can be understood in terms of the eigenvalues of the matrix A in Equation 2.6. By increasing values along the diagonal, ℓ_2 regularization increases eigenvalues of the matrix $(A + \eta I)$ to ensure it is positive definite. Under off-policy distributions, it is possible for A to have eigenvalues that are negative or zero. This implies that there are η for which $\det(A + \eta I) = 0$, and selecting η close to these values allows us to achieve arbitrarily high error. We show one such case in Example 2. This is not merely theoretical–we demonstrate this in the neural network case in Section 2.3.4.

Example 2. When TD is regularized, the model may diverge around (typically small) values of η. Lower-bounding η, a common mitigation, can make well-behaved models vacuous. It is not always possible to select a single value of η that makes models vacuous at different sampling distributions.

Details. Using our three-state example, we set $\mu = [0.05, 0.05, 0.9]$ and solve for $\det(A + \eta I) = 0$:

$$0 = \det(A + \eta I) = \eta^2 + 5.45 \times 10^{-2}\eta - 7.47 \times 10^{-3} \quad \Longrightarrow \quad \eta = 0.0634 \tag{2.26}$$

As in the introductory example, the error grows arbitrarily large as $\eta \to 0.0634$. □

The same analysis is repeated for our second example in 2.3.1. We set $p = 0.9$ and solve for $\det(A + \eta I) = 0$:

Details.

$$0 = 100\eta^2 + 47.4p\eta + 1.85\eta - 1.30p^2 + 0.927p \tag{2.27}$$

$$\eta = 0.00482577 \quad \vee \quad \eta = -0.45 \tag{2.28}$$

Note that the denominator of $g(p, \eta)$ is proportional to $\det(A + \eta I)$, and so $g(0.9, \eta)$ and the error at the TD fixed point can be made arbitrarily large by selecting η close to 4.83×10^{-3}. As this is the only positive root, the model does not diverge at other values. □

This small-η divergence effect can appear in several ways, illustrated in Figure 2.4a.

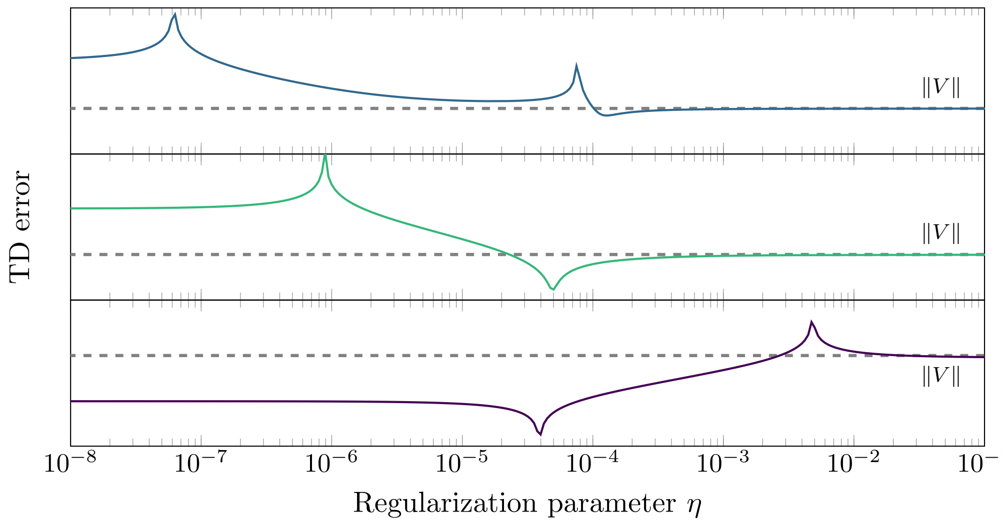
Typically, this appears as one or more points at which TD error diverges before the region at which regularization reduces the model error below $\|V\|$. The first and second plot in Figure 2.4a show two such cases, where the error increases sharply at two and one points respectively.

"Nearly" PSD assumption. In the literature, it is commonly assumed that A is "nearly" positive definite, where only a few eigenvalues are non-positive, and those are close to zero. This gives rise to the common mitigation of setting a lower-bound η_0 such that $(A + \eta I)$ is positive definite for $\eta > \eta_0$. This may render an otherwise well-behaved model vacuous. The third plot in Figure 2.4a illustrates this: the model is not vacuous when unregularized, but is vacuous in the domain $\eta > 10^{-2}$ where divergence is prohibited.

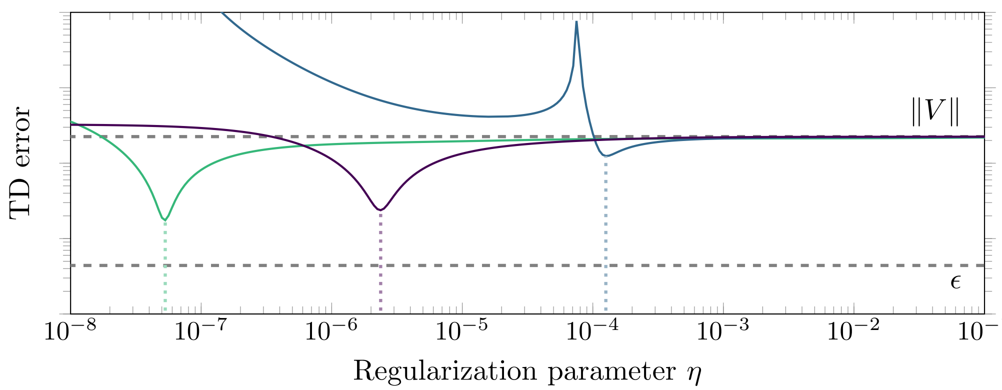
The problem of a fixed η. A common practice in the literature is to set η before training, without regard for the sampling distribution. This is ill advised, as the value may be under- or over-regularizing depending on the sampling distribution. One such example is illustrated in Figure 2.4b, where selecting an η that minimizes the error for one distribution will lead to vacuous or nearly-vacuous results in the other two. A second example in Figure 2.2b has no single η for which trajectories 2 and 3 are both non-vacuous. This is especially relevant as regularization is commonly used to permit distribution drift during training, as discussed in Section 2.4. If the training distribution changes while η is fixed, then algorithms that can be proven to converge to good solutions under some original distribution may converge to poor solutions as the distribution drifts.

2.3.3 Emphatic approaches and our counterexample

Emphatic-TD eliminates instability from off-policy sampling by reweighing incoming data so it appears to be on-policy. There is considerable interest in making this more practical, especially by learning the importance and value models simultaneously. A leading example of this work is COF-PAC [64], which uses ℓ_2-regularized versions



(a) Different MPs at off-policy distributions selected to show small-η error. The error may increase at multiple η, and may even occur *after* the optimal η.



(b) Three off-policy distributions with mutually incompatible η. There is no η at which all models are not vacuous or nearly vacuous.

Figure 2.4: We plot TD error against η to show small-η errors (above) and mutually-incompatible η (below). We also plot the error at the limit of vacuity $\|V\|$ and the representation error ϵ.

of GTD2 [50] to learn both the value and emphasis models. The authors rely on regularization, particularly because the target policy changes during learning. This makes COF-PAC vulnerable to regularization-caused error. We illustrate this with Example 3 in which COF-PAC learns correctly when unregularized, but has large error when regularized.

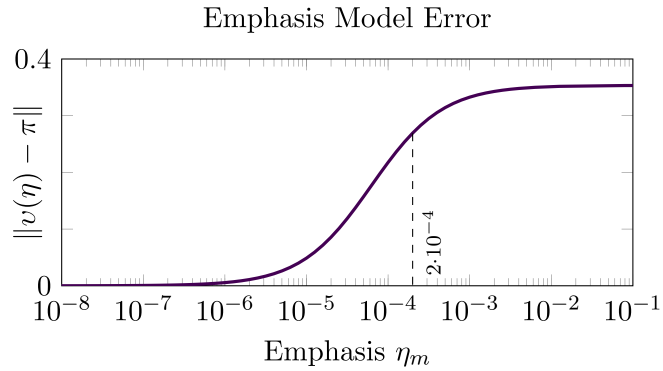
Example 3. Even if unregularized COF-PAC learns the value function with low error, regularizing it may induce arbitrarily large errror.

Details. Conceptually, COF-PAC maintains two separate models that are each updated by TD: the emphasis and the value models. This emphasis model is used to reweigh TD updates so they appear to come from the on-policy distribution. Our strategy is to first show how regularization biases the emphasis model and then how this bias causes the value model to diverge. We begin with our three-state MP, noting its on-policy distribution is $\pi = [.25\ .25,\ .5]$. We wish to learn the values using COF-PAC while sampling off-policy at $\mu = [.2\ .2\ .6]$.

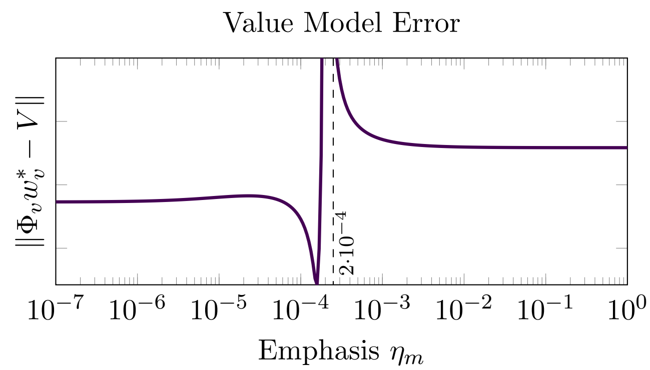
Now we introduce a key conceptual tool: $\upsilon(\eta_m)$, which is the effective distribution seen by the TD-updates, as a function of the emphasis regularization parameter η_m. Unregularized, COF-PAC is able to resample off-policy updates to the on-policy distribution: $\upsilon(0) \equiv \pi$. If the model is regularized, then the effective distribution moves away from π. Figure 2.5a illustrates the distance between $\upsilon(\eta_m)$ and π as the regularization parameter increases.

We can use the effective distribution to compute the error in the value model. Plotting the relationship between the value function error and η_m in Figure 2.5b, we can see the value function has asymptotic error around $\eta_m = 2 \times 10^{-4}$. This shows how COF-PAC may diverge with specific regularization.

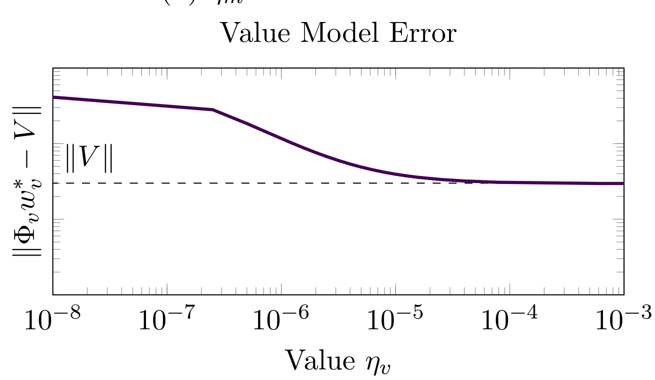
COF-PAC also allows for the value function to be separately regularized with parameter η_v. We show the effect of this in Figure 2.5c, where the value function never does much better than $\|V\|$ making it (nearly) vacuous. We can conclude that regularizing the emphasis model may cause the value model to diverge, and this cannot be fixed



(a) η_m distorts the emphasis model.

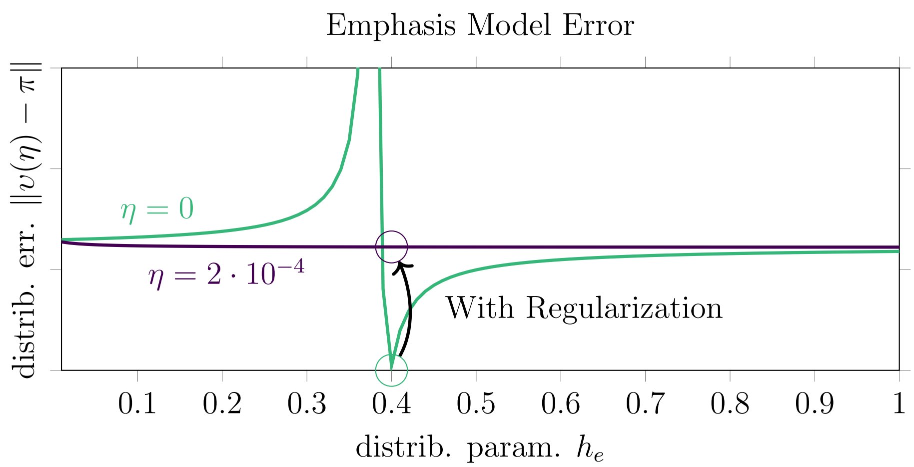


(b) η_m distorts value.

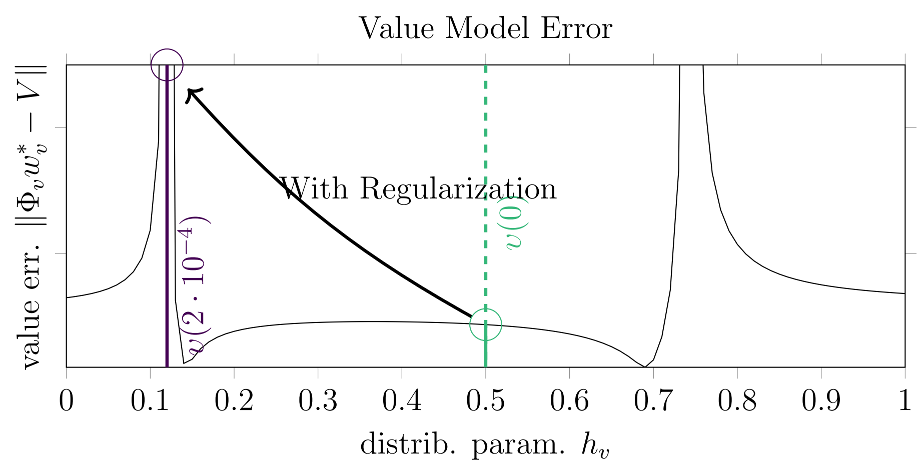


(c) η_v can't fix this.

Figure 2.5: Regularization on the emphasis model (η_m) distorts the effective distribution (Figure 2.5a). Specific values of η_m induce the value function to diverge (Figure 2.5b). The resultant value function is vacuous (Figure 2.5c). Under COF-PAC, regularization can greatly increase model error.



(a) distribution is $[{}^{h_m}/_2,\ {}^{h_m}/_2,\ (1 - h_m)]$



(b) distribution is $[{}^{(1-h_v)}/_2,\ {}^{h_v}/_2,\ 0.5]$

Figure 2.6: Regularization distorts the emphasis model (above), which induces the value function (below) to move to a singularity. Unregularized models are shown in green, regularized models in purple. Regularization can interact with emphasis models to significantly worsen learned value functions.

by regularizing the value function separately. $\square$

Mathematical details of example. We use an MP with the same transition function as in Figure 2.1a, with separate bases Φ_m and Φ_v for the emphasis and value stages respectively. We assume that our interest in all states is uniformly $i = 1$.

We begin by setting the off-policy sampling distribution of $\mu = [.2\ .2\ .6]$, used as the diagonal matrix $D_\mu = \text{diag}(\mu)$. Thanks to the simple structure of our example, we can directly compute the emphasis as $m = \frac{i}{1-\gamma} \cdot \pi D_\mu^{-1} \propto (5/4, 5/4, 5/6)$. We select a basis that allows us to represent this:

$$\Phi_m = \begin{bmatrix} 5/4 & 0 \\ 0 & -1/100 \cdot 5/4 \\ 5/12 & -1/100 \cdot 5/12 \end{bmatrix} \tag{2.29}$$

We deliberately choose Φ_m to have a poor condition number for reasons that will become apparent later. We can represent $c \cdot (5/4, 5/4, 5/6)$ exactly for any constant c:

$$\Phi_m \cdot (1, -100) \cdot c = c \cdot (5/4, 5/4, 5/6) \tag{2.30}$$

Using Equation 5 from [64], we define the matrices:

$$C_m = \Phi_m^\top D_\mu \Phi_m = \begin{bmatrix} 0.417 & -1.04 \times 10^{-3} \\ -1.04 \times 10^{-3} & 4.17 \times 10^{-5} \end{bmatrix} \tag{2.31}$$

$$A_m = \Phi_m^\top (I - \gamma P^\top) D_\mu \Phi_m = \begin{bmatrix} 0.159 & 1.536 \times 10^{-3} \\ 1.536 \times 10^{-3} & 1.59 \times 10^{-5} \end{bmatrix} \tag{2.32}$$

And we apply these to the formulation in Lemma 3 and compute the emphasis weights as a function of the regularization $w_m : \mathbb{R}_0^+ \to \mathbb{R}^+$:

$$w_m^*(\eta) = (A_m^\top C_m^{-1} A_m + \eta I)^{-1} A_m^\top C_m^{-1} \Phi_m^\top D i \tag{2.33}$$

We can then use this to compute the new apparent distribution $\upsilon(\eta)$, which is the effective distribution that the updates to the value model see, and it is equal to the emphasis multiplied by the off-policy distribution.

$$\upsilon(\eta) = \Phi_m \cdot w_m^*(\eta) \cdot D \tag{2.34}$$

Without any regularization, this should be exactly equal to the on-policy distribution.

$$\upsilon(0) = [0.25\ 0.25\ 0.5] \equiv \pi \tag{2.35}$$

When we compute this value with a small amount of regularization $\eta = 2 \times 10^{-4}$, we observe that the apparent distribution drifts far away from the on-policy distribution.

$$\upsilon(2 \times 10^{-4}) = [0.44\ 0.06\ 0.5] \tag{2.36}$$

The proximate cause of this is the poor condition number of C, caused by the $\frac{1}{100}$ scale factor applied to the second column of Φ_m. This allows η to affect different columns by different (relative) amounts in the definition of $w^*(\eta)$, which pushes it away from the symmetric solution. This error shift is visualized in Figure 2.6a.

So far, we have shown how regularization causes a shift in the apparent distribution that the TD updates see. To complete the example we show how this moves the fixed point of the value function away from a stable point into an asymptote where it may grow without bounds. This second phase follows in the same pattern as the first phase, starting with the desired value function: $V = [1\ 2.69\ 1.05]$ and a basis that can nearly[2] represent the value function:

$$\Phi_v = \begin{bmatrix} 1 & 0 \\ 0 & -2.69 \\ {}^1\!/\!{}_2(\epsilon + 1.05) & -{}^1\!/\!{}_2(\epsilon + 1.05) \end{bmatrix} \tag{2.37}$$

$$\epsilon = 2 \times 10^{-4} \tag{2.38}$$

[2]As before, this is necessary as it forces some amount of function approximation.

We use this basis to compute the state-rewards $R = (I - \gamma P)V = [-0.43\ 1.26\ -0.38]$ and define the matrices A_v and C_v and the solution $w_v^*(\eta)$:

$$A_v = \Phi_v^\top (I - \gamma P^\top) D \Phi_v \tag{2.39}$$

$$C_v = \Phi_v^\top D \Phi_v \tag{2.40}$$

$$w_v^*(\eta) = (A_v^\top C_v^{-1} A_v + \eta I)^{-1} A_v^\top C_v^{-1} \Phi_v^\top D R \tag{2.41}$$

We can use this solution to compute the error between the value function and the true values, $\|\Phi_v w_v(\eta) - V\|$. First, under the correctly-resampled distribution without regularization $\upsilon(0) \equiv \pi$:

$$\Phi_v w_v^*(0)|_{D=\mathrm{diag}(\upsilon(0))} = 0.000865 \tag{2.42}$$

Then, with regularization in the value function (but not in the emphasis function):

$$\Phi_v w_v^*(2 \times 10^{-4})|_{D=\mathrm{diag}(\upsilon(0))} = 0.0162 \tag{2.43}$$

Then, under the apparent distribution $\upsilon(2 \times 10^{-4})$ induced by use of regularization in the emphasis function, without and with regularization respectively:

$$\Phi_v w_v^*(0)|_{D=\mathrm{diag}(\upsilon(2\times 10^{-4}))} = 418.601 \tag{2.44}$$

$$\Phi_v w_v^*(2 \times 10^{-4})|_{D=\mathrm{diag}(\upsilon(2\times 10^{-4}))} = 3.00 \tag{2.45}$$

It is immediately obvious that the use of regularization in the emphasis function causes the learned value function to be incorrect. Including a regularizing term in the value estimate is not sufficient to fix the value function. This completes the example.

The non-expansion condition and our counterexample.

COF-PAC makes the strong assumption that Kolter's non-expansion condition [24, eqn. 10] holds in both the emphasis and value models [64, asm. 4]. This is itself a very strong condition because it inherently assumes that both the emphasis and value

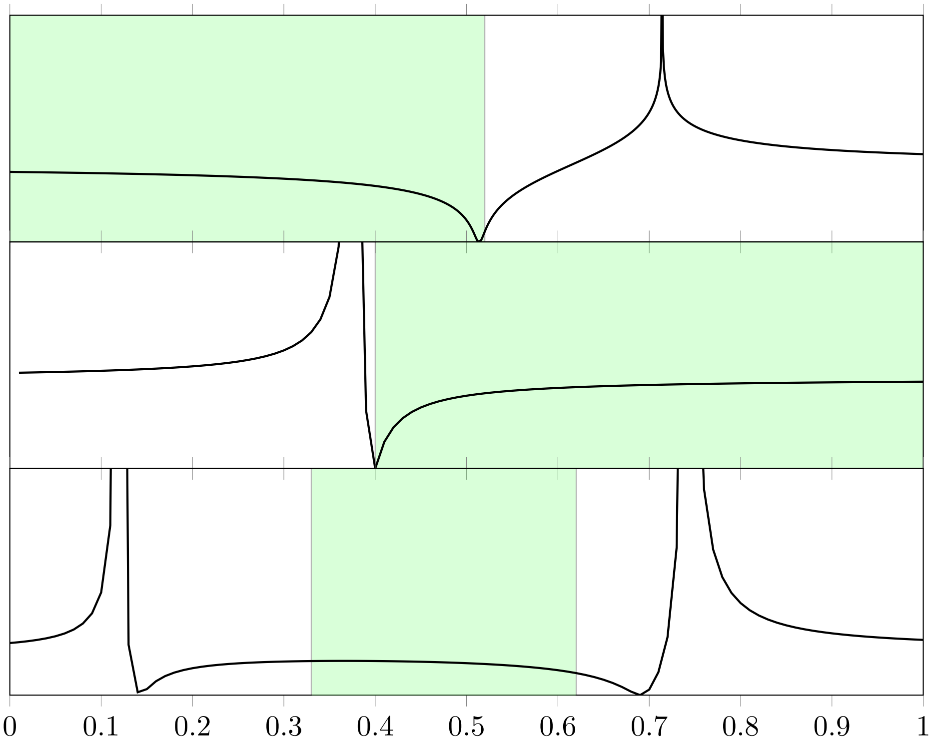


Figure 2.7: The non-expansion condition holds in the shaded region of each graph. These correspond to Figure 2.3, Figure 2.6a, and Figure 2.6b respectively.

models are not subject to runaway TD [64, asm. 4]. This condition selects a convex subset of distributions under which one-step transition followed by projection onto Φ is non-expansive. We illustrate these regions in Figure 2.7. Even in the one-dimensional parameterization shown, this condition only holds in a small sub-region of the space, which suggests that it is a very strong condition.

2.3.4 Applied to multi-layer networks

We use a 9-state variant of our example to study the deadly triad in multi-layer neural networks (NNs). The MDP and its transition function are depicted in Figure 2.9; we have transformed the original MDP by replacing each self-loop with two additional states, forming a clique with the original state. We also define a deterministic

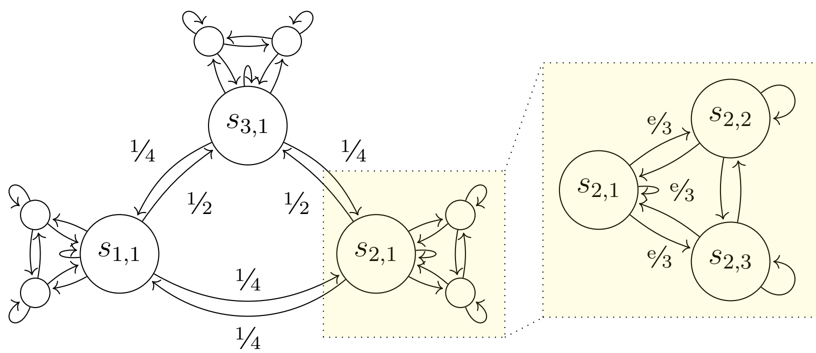


Figure 2.8: Our three-state counter-example MP is extended to nine states to illustrate how the deadly triad problem could manifest in multi-layer neural networks. The self-loop in the original example is replaced with a clique with uniform transitions except as labelled with the original edge weight e.

$$\frac{1}{12}\begin{bmatrix} 1 & 1 & 1 & 3 & & & 6 & & \\ 4 & 4 & 4 & & & & & & \\ 4 & 4 & 4 & & & & & & \\ 3 & & & 1 & 1 & 1 & 6 & & \\ & & & 4 & 4 & 4 & & & \\ & & & 4 & 4 & 4 & & & \\ 3 & & & 3 & & & 2 & 2 & 2 \\ & & & & & & 4 & 4 & 4 \\ & & & & & & 4 & 4 & 4 \end{bmatrix}$$

(a) Transition function of the MDP.

$$\frac{1}{2}\begin{bmatrix} 1 & & & 1 & & \\ & 1 & & 1 & & \\ & & 1 & 1 & & \\ 1 & & & & 1 & \\ & 1 & & & 1 & \\ & & 1 & & 1 & \\ 1 & & & & & 1 \\ & 1 & & & & 1 \\ & & 1 & & & 1 \end{bmatrix}$$

(b) Observation function of the MDP.

Figure 2.9: Our three-state counterexample Markov Process. We use this to illustrate how TD models can fail despite common mitigating strategies with linear function approximation.

observation function $o : \mathcal{S} \to \mathbb{B}^6$. where each state is encoded as the concatenation of the one-hot vector of its subscripts. The value function is assigned pseudo-randomly in range $[-1, 1]$, and a consistent reward function is computed. We select the family of sampling distributions $\mu \propto [4h,\ h,\ h, 4h,\ h,\ h, 8(1-h),\ 4(1-h),\ 4(1-h)]$, where the on-policy distribution is at $h = 0.5$.

We train a simple two-layer neural network with 3 neurons in the hidden layer. The value function is assigned randomly in range $[-1, 1]$.

Example 4. Vacuous models and small-η error also occur in neural network conditions.

Details. We train 100 models using simple semi-gradient TD updates under a fixed learning rate. We plot the mean and the 10^{th}–90^{th} percentile range in Figure 2.10a, with and without regularization. TD is known to exhibit high variance, and regularization is the traditional remedy for that. We corroborate this by noting that the performance of the unregularized model varies widely, but regularization leads to similar performance across initializations at the cost of increased error.

First, we show that vacuous models may exist in the neural network case. In Figure 2.10a, note how there are some off-policy distributions under which both the regularized and unregularized models perform worse than the threshold of vacuity. This is numerical verification that vacuous models exist. Second, we show the small-η error problem in the neural network case in Figure 2.10b, where we plot the TD error against η at a fixed off-policy distribution. We observe that around $\eta \approx 10^{-3}$ the TD Error unexpectedly *increases* before decreasing, which clearly illustrates this phenomenon. $\square$

We wish to learn the model with a two-layer network with $k < n$ nodes in the inner layer. We define the network as $f(o(s_{i,j})) = \tan^{-1}(o(s_{i,j}) * \omega_1) * \omega_2$. The parameters $\omega_1 \in \mathbb{R}^{6\times k}$, $\omega_2 \in \mathbb{R}^{k\times 1}$ are trained to convergence using simple TD updates with semi-gradient updates, a fixed learning rate, and without a target network.

In addition to the example in Figure 2.10b, we present an additional example in

Figure 2.11. The same Markov process, at a different off-policy distribution, attains a curve where the non-vacuous region lies before the divergent region, similar to the second row in Figure 2.4a. An added observation is that these two graphs are mutually incompatible – there is no fixed η that can simultaneously do better than vacuity in both, which promotes the idea of testing multiple regularization parameters or using an adaptive regularization scheme.

2.3.5 Over-parameterization does not solve this problem

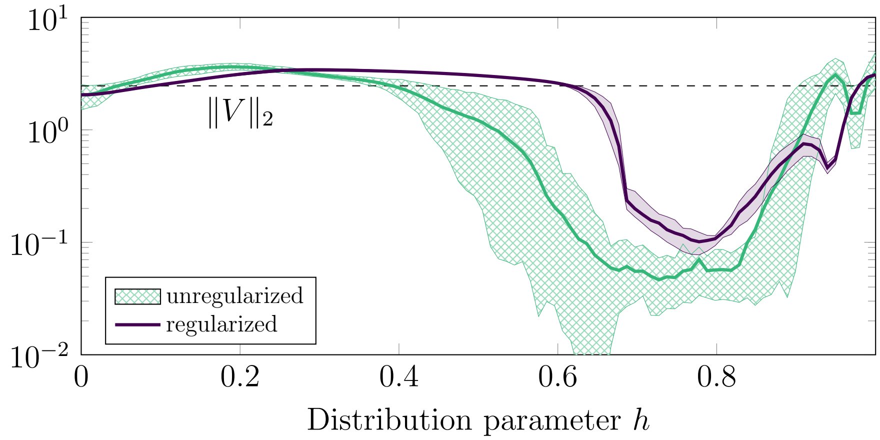
Baird's counterexample [59] shows how, in the linear case, that off-policy divergence can also happen with over-parameterization, as long as some amount of function approximation occurs. It is not obvious that this conclusion persists in the neural network case, so we include an additional example showing that small-η divergence is not solved by over-parameterization.

In Figure 2.12 we plot models with 3 to 13 nodes in the hidden layer. For reference, the MDP has 9 states, so some models under-parameterize and some models over-parameterize. We observe that, in the low-regularization regime, increasing the number of parameters improves the error slightly. However, increasing the number of parameters in the hidden layer does not change the behavior in the the small-η divergence region.

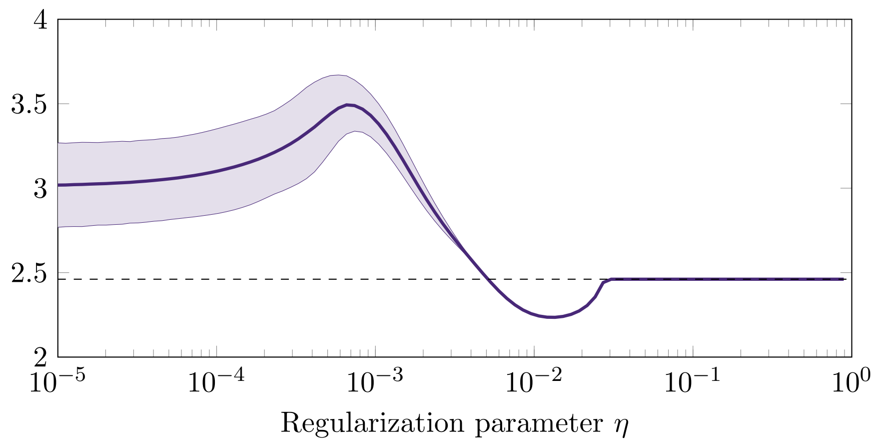
These qualitative links show a clear connection between the neural network case and the linear case, and highlights the importance of correctly handling off-policy sampling.

2.4 Related Work

Three examples of the deadly triad are common in the literature: the classic Tsitsiklis and Van Roy $(w, 2w)$ example [48, p. 260], Kolter's example [24], and Baird's counterexample which shows how training instability can exist despite over-parameterization [59].



(a) Mean and 10^{th}–90^{th} percentile errors of 100 NN value models trained to convergence.



(b) The relationship between error and η at the off-policy distribution $h = 0.31$.

Figure 2.10: We illustrate how regularization interacts with NN value functions, showing that the problems identified in this chapter persist in the NN case.

ℓ_2 regularization is common when proving that an algorithm converges under a changing sampling policy. This is seen in GTD (analyzed in [61]), GTD2 [50], RO-TD [33], and COF-PAC [64]. This assumption may also be used to ensure convergence when training with a target network [63]. Despite the prevalence of regularization, the induced bias from using it is not well studied. It is often dismissed as a mere technical assumption, as in [8]. In this chapter, we contradict that and show how regularization may induce catastrophic bias. By showing concrete examples, this work hopes to inspire further investigation into regularization-induced bias in the same vein as [61].

Alternatives to regularization and TD We focus on ℓ_2 regularization in this chapter, which penalizes the ℓ_2-norm of the learned weights; it is also possible to use ℓ_1 regularization with a proximal operator/saddle point formulation as in [33], or any convex regularization term under a fixed target policy [61]. Instead of directly regularizing the weights, COP-TD uses a discounted update [13]. DisCor [25] propagates bounds on Q-value estimates to quickly converge TD learning in the face of large bootstrapping error; it is not clear if DisCor can overcome off-policy sampling. A separate primal-dual saddle point method has also been adapted to ℓ_2 regularization [9] and is known to converge under deadly triad conditions, and recent work [57] has derived error bounds with improved scaling properties in the linear setting, offering a promising line of research.

Emphatic-TD [49] fixes the fundamental problem in off-policy TD by reweighing updates so they appear on-policy. The core idea underlying these techniques is to estimate the "follow-on trace" for each state, the (weighted, λ- and $\gamma-$discounted) probability mass of all states whose value estimates it influences. This trace is then used to estimate the emphasis, which is the reweighing factor for each update. While this family of methods is provably optimal in expectation, it is subject to tremendous variance in theory and practice, especially when the importance is estimated using Monte-Carlo sampling.[3] In practice, these methods learn the follow-on trace using

[3]Sutton and Barto's textbook [48] says about Emphatic-TD applied to Baird's example that "it is nigh impossible to get consistent results in computational experiments."

TD [19, 64] or similar [17], which makes them vulnerable to bias induced by the use of regularization.

2.5 Relationship to modern RL algorithms

It is still not obvious how strongly this instability affects modern RL algorithms, which are also sensitive to a variety of other failure modes. Unlike our examples, the sampling distribution changes during training, and regularization mechanisms are more complex than simple ℓ_2 penalities. The exact relationship between the instabilities we study and RL algorithms is an open problem, but we offer two pieces of indirect evidence suggesting there is a link.

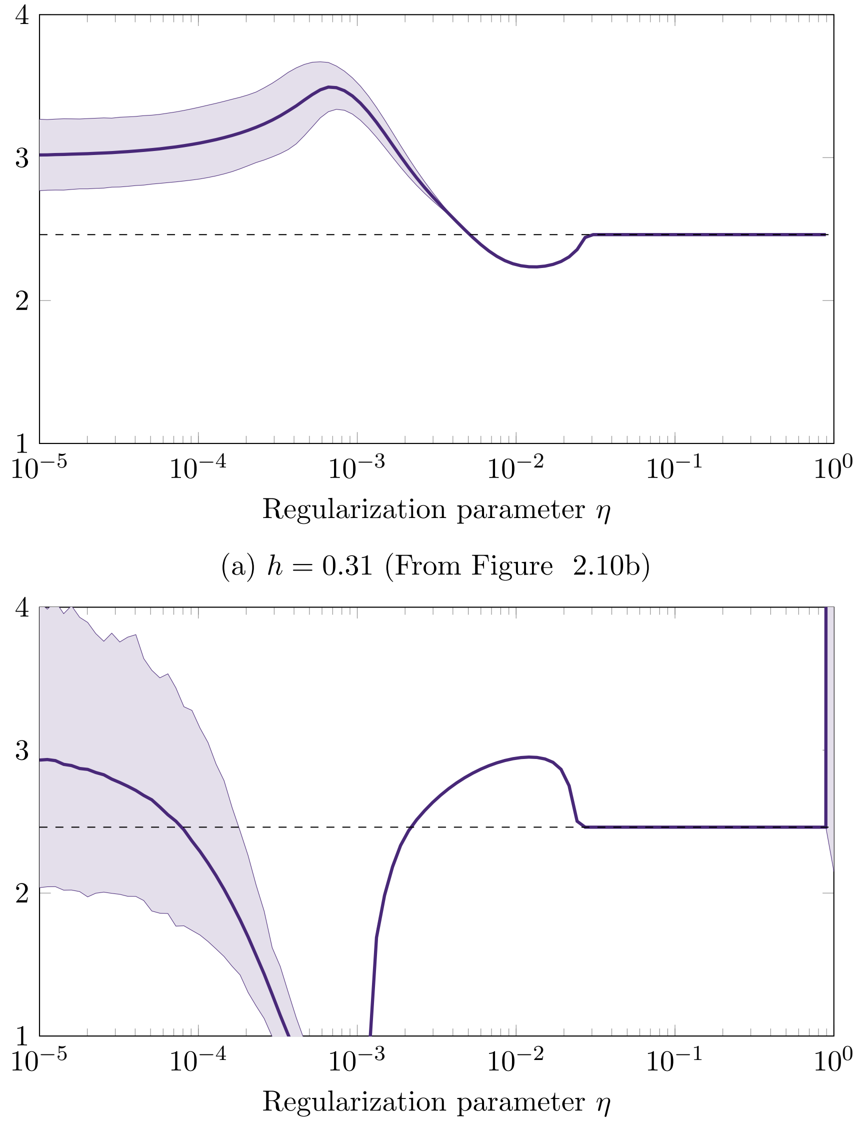
First, in the offline/batch RL literature, it is well-known that online RL algorithms naively applied can catastrophically fail if the learned policy is not consistent with the data distribution. This is known as the distribution shift problem, [31, p. 26] and offline RL algorithms are generally constructed to explicitly address this. Second, when using experience replay buffers in online RL algorithms, policy quality generally improves when older transitions are more quickly evicted [10]. However, there are multiple factors at work here, and it is not possible to cleanly separate the instability from off-policy sampling from the remaining factors.

2.6 Conclusion

There is a tremendous focus in the RL literature on proving convergence of novel algorithms, but not on the error at convergence. Papers like [63] are laudable because they provide error bounds; even if the current bounds are loose, future work will no doubt tighten them. In this work, we show that the popular technique of ℓ_2 regularization does not always prevent singularities and could even introduce catastrophic divergence. We show this with a new counterexample that elegantly illustrates the problems with learning off-policy and how it persists into the NN case.

Even though regularization can catastrophically fail in the ways we illustrate, it remains a reasonable method that may offer a fair tradeoff—as long as we are careful to check that we are not running afoul of the failure modes we explain here. It may be possible to design an adaptive regularization scheme that can avoid these pathologies. For now, testing the model performance over a range of regularization parameters (spanning several orders of magnitude) is the best option we have to detect such pathological behavior.

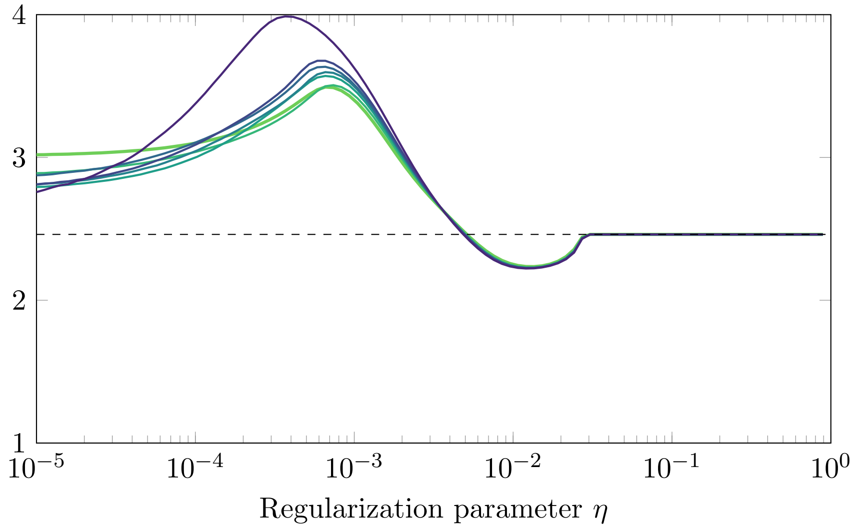
Emphatic-TD is perhaps the most promising area of research for mitigating off-policy TD-learning. The key problem preventing its widespread adoption is the difficulty in estimating the emphasis function, but future work in this area may be able to overcome this. Our example shows the risk of relying on regularization in practical implementations of such methods. It is absolutely critical that Emphatic algorithms correctly manage regularization to avoid the risks that we highlight here.



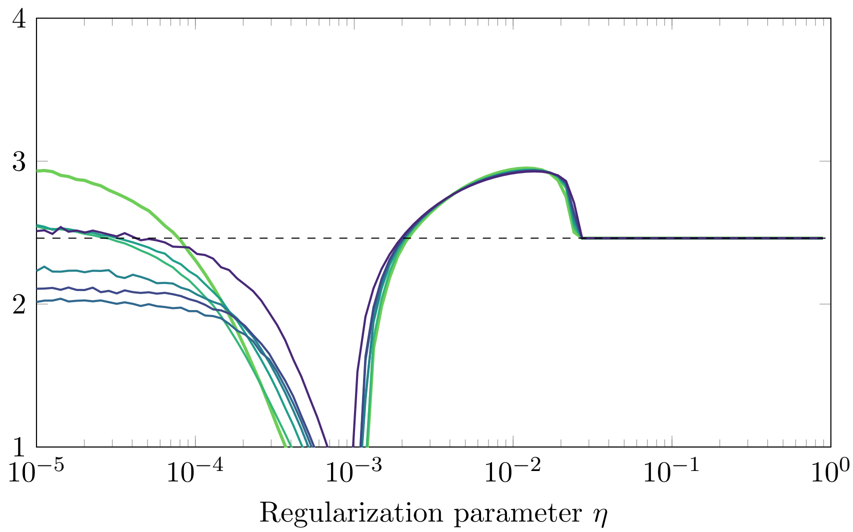
(a) $h = 0.31$ (From Figure 2.10b)

(b) $h = 0.95$

Figure 2.11: The relationship between error and η at different off-policy distributions, showing mutually incompatible regularization behavior. The shaded range indicates the region between the 5th and 95th percentile of 100 differently-initialized models.



(a) $h = 0.31$



(b) $h = 0.95$

Figure 2.12: The relationship between η and error with different amount of model parameterization (with 3, 5, 7, 9, 11, 13, and 64 nodes in the hidden layer, corresponding to darkening colors.)

Chapter 3

Projected Off-Policy TD for Offline Reinforcement Learning

A key problem in offline Reinforcement Learning (RL) is the mismatch between the dataset and the distribution over states and actions visited by the learned policy, called the *distribution shift*. This is typically addressed by constraining the learned policy to be close to the data generating policy, at the cost of performance of the learned policy. We propose Projected Off-Policy TD (POP-TD), a new critic update rule that resamples TD updates to allow the learned policy to be distant from the data policy without catastrophic divergence. Unlike Emphatic-TD and the importance sampling literature, we resample to any "safe" distribution, not necessarily the on-policy. We show how this algorithm works on a well-understood toy example from the literature, and then characterize its performance with varying parameterization on a specially-constructed offline RL task. This is a novel approach to stabilizing off-policy RL, and sets the stage for future work on larger tasks.

Paper in preparation, by Manek, Roderick, and Kolter (2023)

3.1 Introduction

Reinforcement Learning (RL) aims to learn policies that maximize rewards in Markov Decision Processes (MDPs) through interaction, generally using Temporal Difference (TD) methods. In contrast, offline RL focuses on learning optimal policies from a static dataset sampled following an unknown policy, possibly a policy designed for a different task. Thus, algorithms are expected to learn without the ability to interact with the environment. This is useful in environments that are expensive to explore (such as running a Tokamak nuclear reactor [7]), or high-dimensional environments with cheap access to expert or near-expert trajectories (such as video games). Levine et al. [30] present a comprehensive survey of the area.

Since in offline RL the data is gathered before training begins, there is a mismatch between the the state-distributions implied by the learned policy and the data. When applying naive RL algorithms in this setting, they tend to bootstrap from regions with little or no data, causing runaway self-reinforcement. Offline RL algorithms like Conservative Q-Learning (CQL) [26] generally constrain the learned policy to remain within the support of the data. While this works well in practice, there still remains a large gap in performance between online and offline RL. One reason for this is an additional subtlety to distribution shift: because of the combination of off-policy RL and function approximation, it is possible for RL to diverge if the generating policy and the learned policy are sufficiently different even if the data has full support for the learned policy.

We illustrate a simple case in Figure 3.1, where a simple grid environment is designed to elicit the shortest trajectory from start (S) to goal (G). Agents can move one step in each cardinal direction, reaching the goal yields a unit reward, and the episode ends on reaching the goal or any marked cell (X). We generate a dataset by following a suboptimal data policy (···) with sufficient dithering to guarantee that every state-action pair is represented. If we use a tabular Q-function, we can recover the optimal policy (—) and obtain the true value function. When we use a linear Q-function, however, the error is much larger. We find that about half of random initializations

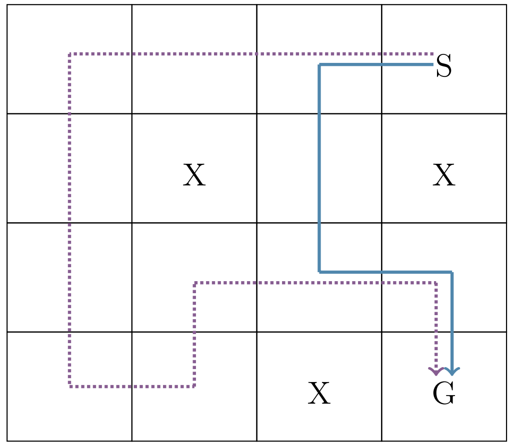


Figure 3.1: A simple grid environment illustrating distribution shift despite complete support. We wish to learn the optimal trajectory (—) from a suboptimal data policy (···) which is ϵ-dithered to get sufficient coverage. When we apply Q-learning methods to this, training often diverges to arbitrarily poor values. This is a consequence of distribution shift. In this paper, we propose a technique to solve this divergence.

lead to Q-functions that either diverge or converge to large error. This shows how even with full coverage of states and actions, distribution shift can be a significant source of error. We provide more details in Section 3.5.1.

In this chapter, we introduce POP-TD, a novel method of mitigating the error from off-policy learning. We show theoretically that this method bounds the off-policy approximation error for TD-based RL methods. We illustrate the resampling process on a well-known toy example, and then demonstrate its effectiveness on an example of offline RL under distribution shift.

3.2 Related Work

Off-Policy TD Learning Instability from learning off-policy has also been studied in the classic RL literature. First described by Tsitsiklis and Van Roy [54], the use of TD learning, function approximation, and off-policy sampling may cause severe instability or divergence. This is known as the *deadly triad* [48, p. 264] and even if many variants of TD still converge, the quality of the solution at convergence may be arbitrarily poor [24].

There are three existing lines of work in the literature that attempt to resolve this: regularization, Emphatic reweighing, and TD Distribution Optimization (TD-DO). The first attempts to regularize TD, typically with $\mathcal{L}_2$-norm weight regularization. Alternative regularization schemes are $\mathcal{L}_1$ [33], convex [61], and bounds propagation [25]. There are well-documented failure modes related to regularization [35]. The second line started with Emphatic-TD, in which Sutton, Mahmood, and White [49] note that it is possible to reweigh samples obtained off-policy so they appear to be on-policy. Such methods learn the follow-on trace using Monte-Carlo methods (in the original), TD [19, 64] or techniques similar to TD [17]. The third method, TD-DO, works by solving a small optimization problem on each TD update to reweigh samples to satisfy the Non-Expansion Criterion, which we introduce in the next section.

Off-Policy and Offline Deep RL Nearly all modern TD-based deep RL methods perform off-policy learning in practice. To improve data efficiency and learning stability, an experience replay buffer is often used. This buffer stores samples from previous versions of the policy [38], and so the distribution of the data is not on-policy for the current version of the policy. Additionally, exploration policies, such as a epsilon greedy [48, p. 100] or Soft Actor Critic (SAC)-style entropy regularization [15] [1], are often used, which also results in off-policy learning. In practice, the difference between the current policy and the samples in the buffer is limited by setting a limit to the buffer size and discarding old data; or by keeping the exploration policy relatively close to the learned policy. In practice, this is sufficient to prevent outright divergence, though the extent to which it decreases performance is not well-understood.

However, in the offline RL setting where training data is static, there is usually a much larger discrepancy between the state-action distribution of the data and the distribution induced by the learned policy. This discrepancy presents a significant

[1] While the original SAC algorithm is technically on-policy since it learns an entropy-regularized value function, the entropy-regularization is often dropped from the value-function estimate in practice to improve performance.

challenge for offline RL [30]. While this distributional discrepancy is often presented as a single challenge for offline RL algorithms, it is convenient to separate the two distinct aspects of this challenge and address them independently: *support mismatch* and *proportional mismatch*. When the support of the two distributions differ, learned value functions will have arbitrarily high errors in low-data regions. Support mismatch is dealt with by either constraining the KL-divergence between the data and learned policies [11, 28, 60], by penalizing or pruning low-support (or high-uncertainty) actions [26, 62, 22].

Even when the support of the data distribution matches that of the policy distribution, naive TD methods can produce unbounded errors in the value function [54]. We call this challenge *proportional mismatch*.

Importance sampling (IS) [44] is one of the most widely used techniques to address proportional mismatch. The idea with IS is to compute the differences between the data and policy distributions for every state-action pair and re-weight the TD updates accordingly. However, these methods suffer from variance that grows exponentially in the trajectory length. Several methods have been proposed to mitigate this challenge and improve performance of IS in practice [16, 13, 40, 39, 32], but the learning is still far less stable than other offline deep RL methods. In this work, we propose a new method to bound the value-function approximation errors caused by proportional mismatch without the need to explicitly compute (or approximate) IS weights.

3.3 Problem Setting and Notation

Consider the n-state Markov chain $(\mathcal{S}, P, R, \gamma)$, with finite state space $\mathcal{S}$, transition function $P : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_+$, reward function $R : \mathcal{S} \to \mathbb{R}$, and discount factor $\gamma \in [0, 1]$. Because the state-space is finite, it can be indexed as $\mathcal{S} = \{1, \ldots, n\}$. This allows us to use matrix rather than operator notation. The expected γ-discounted future reward of being in each state $V(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \middle| s_0 = s\right]$ is called the value function. The value function is consistent with Bellman's equation (in matrix form):

$$V = R + \gamma P V \tag{3.1}$$

In the linear setting, we approximate the value function as $V(s) \approx w^\top \phi(s)$, where $\phi : \mathcal{S} \to \mathbb{R}^k$ is a fixed basis function and we estimate parameters $w \in \mathbb{R}^k$. In matrix notation, we write this as $V \approx \Phi w$.

In this work, we are interested in the offline learning setting, where the sampling distribution μ differs from the stationary distribution ν. In this setting, the previous equation is insufficient. We need to account for the function approximation, and so the TD solution is:

$$\Phi w = \Pi_\mu (R + \gamma P \Phi w) \tag{3.2}$$

where $\Pi_\mu = \Phi(\Phi^\top D_\mu \Phi)^{-1} \Phi^\top D_\mu$ is the projection onto the column space of Φ weighted by the data distribution μ through the matrix $D_\mu = \operatorname{diag}(\mu)$. This projection may be arbitrarily far from the true solution, and so the error may be correspondingly large. The literature bounds the error as:

Theorem 2. *The error at the TD fixed point is $\|\Phi w - V\|_{D_\mu}$. Lemma 6 from [54] bounds this in terms of error projecting V onto the column space of Φ:*

$$\|\Phi w - V\|_{D_\mu} \leq \frac{1}{1-\gamma} \, \|\Pi_\mu V - V\|_{D_\mu} \tag{3.3}$$

3.3.1 The Non-Expansion Criterion (NEC)

Thus far we have left open the notion of a "safe" distribution to resample TD updates to. The on-policy distribution is always safe, but we need to establish some criterion for "safe" off-policy distributions. Tsitsiklis and Van Roy lay the groundwork for this by analyzing the training of on-policy TD as a dynamical system and showing that once TD reaches its fixed point, subsequent TD updates form a non-expansive mapping around that fixed point (1996, lemma 4), and therefore prove that on-policy TD does not diverge.

To do this, they begin with the fact that error bounds from on-policy TD follow the property that the $D-$norm of any vector $x \in \mathbb{R}^n$ is non-expansive through the transition matrix. That is: $\|Px\|_D \leq \|x\|_D$, where $D = \mathrm{diag}(\pi)$. Kolter [24] extend this analysis to the off-policy case, deriving a linear matrix inequality (LMI) under which the TD updates are guaranteed to be non-expansive around the fixed point. This is the Non-Expansion Criterion (2011):

$$\|\Phi w - V\|_D \leq \frac{1 + \gamma\kappa(D^{-1/2}D^{1/2})}{1-\gamma}\|\Pi_D V - V\|_D \tag{3.4}$$

From this bound, he derives the *non-expansion criterion* (NEC):

$$\|\Pi_D P \Phi w\|_D \leq \|\Phi w\|_D \qquad (\forall w \in \mathbb{R}^n) \tag{3.5}$$

This holds if and only if the matrix F_D is positive semi-definite

$$F_D \equiv \begin{bmatrix} \Phi^\top D \Phi & \Phi^\top D P \Phi \\ \Phi^\top P^\top D \Phi & \Phi^\top D \Phi \end{bmatrix} \succcurlyeq 0 \tag{3.6}$$

Equivalently, in terms of the expectation over states:

$$\mathbb{E}_{s\sim\mu, s'\sim p(\cdot|s)} \left[\begin{bmatrix} \phi(s)\phi(s)^\top & \phi(s)\phi(s')^\top \\ \phi(s')\phi(s)^\top & \phi(s)\phi(s)^\top \end{bmatrix} \right] \succcurlyeq 0. \tag{3.7}$$

This constraint describes a convex subset of D. As a $2k \times 2k$ matrix (where k is the number of features), F is prohibitively large to enumerate for any real RL problem, and so our algorithm is designed to make use of this without ever constructing it directly. Further, we notice that the construction of F_D depends on P, the transition matrix of the underlying Markov process, which makes our algorithm more complex.

For convenience, we write this as:

$$\mathbb{E}_{s\sim q}[F(s)] \succcurlyeq 0, \text{ where} \tag{3.8}$$
$$F(s) = \mathbb{E}_{s'\sim p(s'|s)}\left[\begin{bmatrix} \phi(s)\phi(s)^\top & \phi(s)\phi(s')^\top \\ \phi(s')\phi(s)^\top & \phi(s)\phi(s)^\top \end{bmatrix}\right]$$

NEC is an expectation over some state distribution q and transition distribution $p(s, s') = p(s'|s)\mu(s)$. Because it is an LMI, the satisfying state distributions q form a convex subset.

Directly constructing $F(s)$ or $F(s, s')$ is impossible on all but the simplest examples – it would take $\mathcal{O}(k^2 n)$ or $\mathcal{O}(k^2 n^2)$ memory respectively to hold all the necessary data. Instead we exploit the structure inherent in the problem to make use of $F(s)$ without creating it.

3.4 Projected Off-Policy TD (POP-TD)

We propose an alternative approach to stabilizing off-policy training, based on the NEC by Kolter [24]. POP-TD identifies a convex set of "safe" distributions that satisfy NEC and reweighs TD updates to come from that set. In contrast to TD-DO, POP-TD solves a different optimization problem using a two-timescales update with fixed cost per iteration, allowing it to scale to real-world problems.

We begin by deriving the projected off-policy update for Markov Chains without a separate policy function. We will extend this derivation to support actions and Markov Decision Processes (MDPs) in Section 3.4.5. Our algorithm resamples TD updates so they come from some distribution q for which the NEC holds. Given input data $(x_1, x_2, \ldots)$, this is the same as finding a set of weights $q_1, q_2, \ldots$ such that

$$\sum_i q_i \cdot F(x_i) \succcurlyeq 0 \tag{3.9}$$

3.4.1 I- and M-projections

The Kullback-Leibler divergence is an *asymmetric* measure, and so it is usually the case that $\min_q \ \mathrm{KL}(q||\mu) \neq \min_q \ \mathrm{KL}(\mu||q)$. The former ("from μ to q") is an information (or I-)projection, which tends to under-estimate the support of q potentially excluding possible sampling distributions to reweigh to. The latter ("from q to μ") is a moment (or M-)projection, which tends to over-estimate the support of q and avoid zero solutions. In our solution, we are proposing using an I-projection instead of the M-projection used by Kolter [24].

3.4.2 Optimizing the distribution

In the previous section we have characterized a convex subset of off-policy distributions under which TD learning is guaranteed not to diverge. If we can discover any such distribution for a particular TD problem, we can reweigh our TD updates (from any distribution) so they appear consistent with this reweighing distribution. This is related to the main insight in Emphatic-TD [49], with the key innovation that we can take any non-expansive distribution *not just the on-policy distribution*.

We can now write down the optimization problem that we wish to solve:

$$\underset{q}{\text{minimize}} \ \mathrm{KL}(q||\mu) \qquad \text{s.t.} \quad E_{s\sim q}[F(s)] \succcurlyeq 0 \tag{3.10}$$

We are searching for q, the closest distribution to the sampling distribution μ such that F is PSD under q. Note that we could in principle minimize any notion of "closest" to find some satisfying distribution – for example Kolter [24] explores the effects of minimizing $\mathrm{KL}(\mu||q)$.

We construct the dual of this problem:

$$\underset{Z\succcurlyeq 0}{\text{maximize}} \ \underset{q}{\text{minimize}} \ \mathrm{KL}(q||\mu) - \mathrm{tr} Z^{\top}\mathbb{E}_{s\sim q}[F(s)] \tag{3.11}$$

Using the Lagrange multiplier $Z \in \mathbb{R}^{2k\times 2k}$, we solve the inner optimization problem:

$$\underset{q}{\text{minimize}} - H(q) - \mathbb{E}_{s\sim q}[\log \mu(s) + \text{tr}Z^\top F(s)] \tag{3.12}$$

Writing down Lagrangian and solving for the optima, we obtain:

$$q^*(s) \propto \mu(s) \exp(\text{tr}Z^\top F(s)) \tag{3.13}$$

(Subject to the normalization constraint that $\sum_{s\in\mathcal{S}} q^*(s) = 1$.)

Plugging this back into our dual formulation, we obtain the optimization problem:

$$\underset{Z\succcurlyeq 0}{\text{maximize}} - \log \mathbb{E}_{s\sim\mu}[\exp(\text{tr}Z^\top F(s))] \tag{3.14}$$

Which we can simplify to

$$\underset{Z\succcurlyeq 0}{\text{minimize}}\ \mathbb{E}_{s\sim\mu}[\exp(\text{tr}Z^\top F(s))] \tag{3.15}$$

As discussed earlier, $F(s)$ cannot be directly constructed; instead, we assume that Z holds a specific structure and optimize the problem.

3.4.3 The structure of Z

Our next goal is to transform this constrained optimization problem into an unconstrained problem over a low-rank version of Z, suitable for learning via SGD.

We assume (and later check!) that the solution for Z is low-rank. Intuitively, this is because $\mathbb{E}_{s\sim\mu}[F(s)]$ is PSD when μ is close to π, and for most MDPs, sampling off-policy leads to only a small number of negative eigenvalues that need to be corrected by Z. Kolter [24] provides a technical explanation: by the KKT conditions, Z will have rank complementary to $\mathbb{E}_{s\sim\mu}[F(s)]$, and the latter is expected to be full rank. It is worth noting that this "almost-PSD" assumption is common in the field.

We make the mild assumption that Z has rank m, where $m << k$. We apply the Burer-Montiero approach [4] to convert the constrained optimization problem over Z into an unconstrained optimization over low-rank matrices $A \in \mathbb{R}^{k\times m}$ and $B \in \mathbb{R}^{k\times m}$:

$$Z^\star = \begin{bmatrix} A \\ B \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}^T \tag{3.16}$$

This allows us to represent the rank-m PSD matrix Z^* in terms of the unconstrained matrices A and B. Substituting this into the dual formulation, we get:

$$\underset{A,B}{\text{minimize}} \;\; \mathbf{E}_{s\sim\mu}\left[\exp\left(\text{tr} \begin{bmatrix} A \\ B \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}^T F(s)\right)\right] \tag{3.17}$$

We can leverage the structure of $F(s)$ to simplify the trace term:

$$\begin{aligned}
&\text{tr} Z^T F(s) && (3.18)\\
&= \text{tr} \begin{bmatrix} A \\ B \end{bmatrix}^T \begin{bmatrix} A \\ B \end{bmatrix}^T F(s) && (3.19)\\
&= \text{tr} \begin{bmatrix} A \\ B \end{bmatrix}^T F(s) \begin{bmatrix} A \\ B \end{bmatrix} && (3.20)\\
&= \text{tr} \begin{bmatrix} A \\ B \end{bmatrix}^T \mathbf{E}_{s'\sim p(s'|s)} \begin{bmatrix} \phi(s)\phi(s)^T & \phi(s)\phi(s')^T \\ \phi(s')\phi(s)^T & \phi(s)\phi(s)^T \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} && (3.21)\\
&= \text{tr}\left[(A+B)^T\phi(s)\phi(s)^T(A+B) - 2B^T\mathbf{E}_{s'\sim p(s'|s)}\left[\phi(s)(\phi(s)-\phi(s'))^T\right]A\right] && (3.22)\\
&= \|(A+B)^T\phi(s)\|^2 - \text{tr}\left[2B^T\mathbf{E}_{s'\sim p(s'|s)}\left[\phi(s)(\phi(s)-\phi(s'))^T\right]A\right] && (3.23)
\end{aligned}$$

This allows us to rewrite the optimization problem as:

$$\underset{A,B}{\text{minimize}} \;\; \mathbf{E}_{s\sim\mu}\left[\exp\left(\begin{array}{l} \|(A+B)^T\phi(s)\|^2 \\ \qquad -\text{tr}\left[2B^T\mathbf{E}_{s'\sim p(s'|s)}\left[\phi(s)(\phi(s)-\phi(s'))^T\right]A\right] \end{array}\right)\right] \tag{3.24}$$

where the small parameters A and B can be optimized with regular gradient-descent methods.

3.4.4 Update rules

We can't directly optimize our problem because that would require us to estimate the inner expectation term. Instead, we use a two-timescales approach by estimating two mutually-dependent quantities separately and improving them at potentially different rates. This generally converges to a valid solution with a little tuning.

We choose to estimate the matrices $A, B \in \mathbb{R}^{k \times m}$ and separately the function $g_\theta : \mathcal{S} \in \mathbb{R}$ where

$$g_\theta(s) \approx \mathrm{tr} Z^T F(s) \tag{3.25}$$

which can be approximated as a linear function (or a neural network) with parameters θ. The size of the weights learned by POP-TD are therefore $\mathcal{O}(k)$, comparable to the size of vanilla Q-learning.

This corresponds to the auxiliary loss term for A, B:

$$\mathcal{L}_{A,B}(s, s') = \exp(g_\theta(s)) \left[\|(A+B)^T \phi(s)\|^2 - \mathrm{tr}\left[2B^T \phi(s)(\phi(s) - \phi(s'))^T A \right] \right] \tag{3.26}$$

and for g:

$$\mathcal{L}_g(s, s') = \left(g_\theta(s) - \left[\|(A+B)^T \phi(s)\|^2 - \mathrm{tr}\left[2B^T \phi(s)(\phi(s) - \phi(s'))^T A \right] \right] \right)^2 \tag{3.27}$$

And finally, when updating the value function weights w, we multiply the loss associated with each transition by $\exp(g(s))$ to resample it so it appears to come from the "safe" distribution, which completes the description of the algorithm!

Computing the loss A naive implementation of the loss function will require intermediate matrices of size $[k \times k]$. We can improve speed by computing the loss in terms of $[m \times 1]$ intermediates instead. For a transition sample (s, s'), this can be done as:

$$
\begin{aligned}
M_A &= A^T \phi(s) \in \mathbb{R}^m \\
M'_A &= A^T \phi(s') \in \mathbb{R}^m \\
M_B &= B^T \phi(s) \in \mathbb{R}^m \\
\mathcal{L}_{A,B}(s, s') &\equiv \exp(g_\theta(s)) \left[\|M_A\|^2 + \|M_B\|^2 + 2M'_A \cdot M_B\right] \qquad (3.28) \\
\mathcal{L}_g(s, s') &\equiv \left(g_\theta(s) - \left[\|M_A\|^2 + \|M_B\|^2 + 2M'_A \cdot M_B\right]\right)^2 \qquad (3.29)
\end{aligned}
$$

where $\cdot$ is the dot product. With tabular g, this sequence of operations should be $\mathcal{O}(mk)$, which is much quicker than the naive $\mathcal{O}(mk^2)$.

3.4.5 POP-Q-Learning

Thus far, we have focused on Markov Reward Processes. For RL problems, we need to extend this approach to Markov Decision Processes (MDPs). An MDP is a tuple, $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, with state space $\mathcal{S}$, probabilitistic transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$, reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and discount factor $\gamma \in [0, 1]$. The goal in this setting is to find a probabilistic policy $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_+$ that maximizes the future discounted reward:

$$\pi^\star = \arg\max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \tag{3.30}$$

Many RL methods use some variation of Q-learning [58, 38, 15, 26], which involves learning a state-action value function commonly called a Q-function:

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| s_0 = s, a_0 = a\right] \tag{3.31}$$

By considering a fixed policy π, a combined state-space $\mathcal{X} = \mathcal{S} \times \mathcal{A}$, and a policy-conditioned transition function $\tilde{P}^\pi((s, a), (s', a')) = P(s, a, s')\pi(s', a')$, any MDP

Algorithm 1 Deep POP-Q-Learning

Initialize Q-function, Q_{θ^Q}, g-function, g_{θ^g}, dual variable vector y, and some policy π_{θ^π}.
for step t in $1, \ldots, N$ **do**
  Sample mini-batch $(s, a, r, s') \sim \mu$.
  Sample $\tilde{a} \sim \pi_{\theta^\pi}(s), \tilde{a}' \sim \pi_{\theta^\pi}(s')$.
  # Compute features from penultimate layer of Q-network:
  $\phi \leftarrow Q_{\theta^Q}(s, a), \phi' \leftarrow Q_{\theta^Q}(s', \tilde{a}')$.
  # Update g-function and dual variable vectors:
  $\theta^g_t \leftarrow \theta^g_{t-1} - \eta_g \nabla_{\theta^g} \mathcal{L}_g(s, s')$
  $A_t \leftarrow A_{t-1} - \eta_A \nabla_A \mathcal{L}_A(s, s')$
  $B_t \leftarrow B_{t-1} - \eta_B \nabla_B \mathcal{L}_B(s, s')$
  # Update Q-function using re-weighted Q-loss update:
  $\theta^Q_t \leftarrow \theta^Q_{t-1} - \eta_Q \exp(g_{\theta^g}(s, a)) \nabla_{\theta^Q} \mathcal{L}_Q(\theta^Q)$
  # Update policy with SAC-style loss:
  $\theta^\pi_t \leftarrow \theta^\pi_{t-1} - \eta_\pi \nabla_{\theta_\pi} [Q_{\theta^Q}(s, \tilde{a}) - \log \pi_{\theta^\pi}(\tilde{a}|s)]$
end for

reduces to a Markov Chain. Thus, as long as the NEC is satisfied in this modified state-space, we can bound the approximation error of the Q-function. See Section 3.4.5 for a detailed derivation.

Finally, for our method to applied to modern deep RL problems, we must extend our approach to non-linear Q-functions. To do so, we approximate the Q-function with a neural network, Q_{θ^Q} parameterized by θ^Q and consider a stochastic parameterized policy π_{θ^π}. To update Q_{θ^Q}, we used a squared Bellman loss, $\mathcal{L}_Q(\theta^Q) = (Q_{\theta^Q}(s, a) - r - \gamma Q_{\theta^Q}(s', \pi_{\theta^\pi}(s')))^2$, which we reweigh with $\exp(g(s))$ as before. For our offline RL experiments, we also add CQL regularization [26] to our Q-learning updates to prevent over-optimism on low-support regions of the state-action space. To update our linear dual variables y, we use the penultimate layer of Q_{θ^Q} as our feature vector. Finally, we use a SAC-style entropy regularized loss to update our policy network, π_{θ^π}. Algorithm 1 provides an overview of our method.

3.5 Experiments and Discussion

We first apply POP-TD to a well-understood example so that we can directly illustrate the how it resamples TD updates to a "safe" distribution. We use the simple three-state task from Figure 3.2, including the specified transition function, value function, and basis. Since this is a policy evaluation task, there is no policy to be separately learned.

For illustration purposes, we select the family of distributions $\pi = ({}^{h}\!/_{2}, {}^{h}\!/_{2}, 1 - h)$ parameterized by $h \in [0, 1]$. This characterizes the possible distributions of data that we will present to POP-TD and naive TD in this experiment. The on-policy distribution corresponds to $h_o \approx 0.51$, and divides the family of distributions into a left subset ($h \leq h_o$) where the NEC holds and a right subset ($h > h_o$) where it does not. This is immediately apparent in Figure 3.2, where we plot the error at convergence from running naive- and POP-TD above, and the effective distribution of TD updates after reweighing below. In the left subset, where the NEC holds, POP-TD does not resample TD updates at all. Therefore, the error of POP-TD tracks naive TD (top), and the effective distribution of TD updates in POP-TD and naive TD are the same as the data distribution (bottom).

In the right subset, we observe that naive TD converges to poor solutions with large error while POP-TD is able to learn with low error. Directly computing the effective distribution, we see that naive TD adheres to the data distribution but POP-TD resamples the TD updates. Looking at the behavior of POP-TD in the right subset, we see that POP-TD resamples updates to the on-policy distribution p_o in $p \in [p_o, 0.9]$, corresponding to the horizontal segment. This allows the learned Q-function to have very low error in that domain. As the data distribution becomes more extreme ($p \in [0.9, 1)$), POP-TD is not quite able to learn the resampling ratio, and so the effective distribution shifts away from p_o. This leads to a corresponding slight increase in error at extreme ratios. From this we observe that POP-TD requires full support of the sampling distribution, similar to many offline RL algorithms [26, 47].

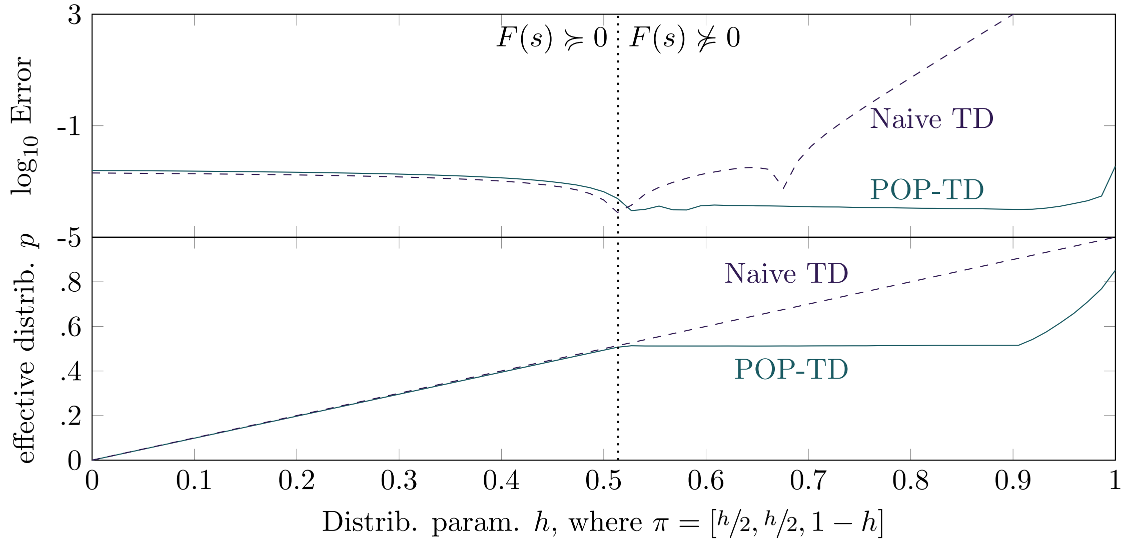


Figure 3.2: The error in the learned value function by naive- and POP-TD, plotted against a varying sampling distribution. In the left half of the plot, the NEC holds, and so POP-TD tracks the error of naive TD closely. In the right half of the plot naive TD diverges, while POP-TD resamples the data to a "safe" distribution and does not diverge.

This simple experiment cleanly illustrates how POP-TD resamples TD updates to come from a "safe" distribution, and how that can greatly reduce the error in a policy evaluation task.

3.5.1 POP-Q on GridWorld

In this experiment, we consider the the simple grid environment from Figure 3.1, modified to add transitions from terminating states to the starting state. Our goal is to approximate the true Q-function with minimal error. Our training data is sampled following the suboptimal data policy (···), adding uniform random dithering to guarantee that every state-action pair is represented. We represent the Q-function as a linear function with a fixed random basis $\Phi \in \mathbb{R}^{64\times 53}$, training it to convergence using naive Q-learning and linear POP-Q separately. For POP-Q, we also randomly initialize matrices $A, B \in \mathbb{R}^{53\times 4}$ and a tabular $g \in \mathbb{R}^{64}$ separately. (We will later

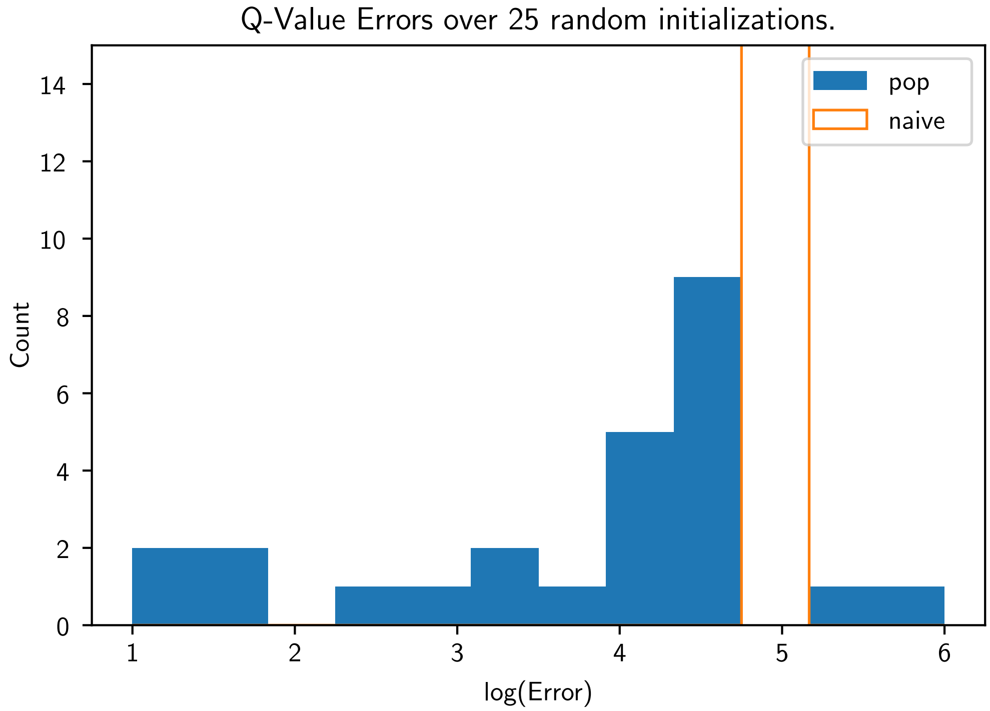


Figure 3.3: Log Q-function errors for naive and POP Q-Learning on Figure 3.1, over 25 randomly sampled bases. Errors are computed using a tabular g function, and bins are exponentially wide. POP-Q substantially reduces error in most of the sampled bases.

consider an approximate g.)

Setting the rank of A, B: We could simply set the rank of A and B as any other hyperparameter, but since this problem is sufficiently small we can instead compute the minimum rank directly. To do this, we compute the degree of rank deficiency of the matrix $\mathbb{E}_s[F(s)]$ from Equation (3.8) on our dataset, and set the rank of A and B so the sum of the rank of $\mathbb{E}_s[F(s)]$ and A and B is at least k. For this example, for $k = 53$, we find that $\text{rank}(A) = \text{rank}(B) = 4$ is sufficient for this example.

Results with an exact, tabular g

Since tabular Q-learning always converges to the global optimum [58], we use that to compute the ground-truth Q-function. All error reported is relative to that assumed ground truth.

Figure 3.3 shows the distribution of errors achieved by vanilla and POP Q-learning over 25 different bases on our task. Vanilla Q-learning performs consistently poorly, achieving a (large) amount of error at all seeds. This is expected because we have deliberately engineered the task to be unstable. In comparison, POP-TD improves performance over most seeds, and in some cases enables near-perfect fitting of the Q-function.

Throughout this chapter we have drawn a distinction between importance sampling/Emphatic TD methods and our work. While the former attempts to resample to the on-policy distribution, our work seeks to resample to the closest stable distribution. We illustrate this difference in Figure 3.4, where we display the rates at which states are visited in our GridWorld. The distribution in our dataset (top-right) is far from the on-policy distribution (top-left), which is what importance sampling and Emphatic methods will attempt to resample to. In comparison, POP-Q resamples minimally (bottom row), where the effective distribution reached is very close to the data distribution.

Results with an approximate, linear g

The current experiments with POP-Q learning all use a tabular g. This works, but takes the same memory as would learning a tabular value function, which would provably converge to the global optimum (side-stepping the entire problem). A key step in adopting POP-Q is ensuring that all parameters are at most order $\mathcal{O}(k)$ (i.e. comparable to the size of the learned weights) and are therefore learnable with the same order of time and space as regular TD. We also wish to (eventually) exploit the generalization afforded to us by neural networks, to hopefully learn more accurate models with less data.

The matrices A and B are sized $k \times m$, where $m << k$, and so are sufficiently small. We now need to approximate g as a linear function with fixed bases vectors $\Phi_g = (\phi_{g,1}, \phi_{g,2}, ..., \phi_{g,n})$, $\phi_{g,\circ} \in \mathbb{R}^l$ and learned weights $w_g \in \mathbb{R}^l$ of size $l < n$:

$$g(s) = \phi_{g,s} \cdot w_g \tag{3.32}$$

To understand the relationship between the degree of approximation (as measured by the size of the basis l) and the performance of our system, we initialize 25 different $n \times n$ bases and report the performance as the bases are truncated down from 64 to 1. This is illustrated in Figure 3.5.

Figure 3.5 reveals that the performance of POP-Q is (as expected) sensitive to the exact basis chosen. For some bases, the error increases with only a small amount of approximation, but for some "lottery-ticket" bases, this continues to work even as the bases are truncated to rank 1. For some bases, this continues to work despite extreme approximation is because the degree of resampling required is minimal and the system is fairly easy to resample.

A note on initialization When performing experiments, we note that the performance of POP-TD depends sharply on the condition number of Φ, but not necessarily that of Φ_g. Specifically, we see that an orthogonal initialization step on Φ is crucial for performance. (In this step we set Φ to the orthogonal matrix of the QR-decomposition of a matrix where entries are sampled uniformly at random.) We conjecture that this happens because POP-TD seeks to stochastically learn $\Phi^T A \Phi$, and a poor condition number of Φ leads to values that span multiple orders of magnitude and linear approximation is known to perform poorly on such data.

3.6 Conclusion

In this chapter we introduced POP-TD, a method for effective TD learning under off-policy distributions, with applications to offline RL and learning under large distribution shifts. Unlike existing emphatic TD and importance sampling methods which resample to the on-policy distribution, POP-TD resamples to the closest distribution under which TD will provably not diverge.

We present POP-TD on an existing "deadly triad" example in the literature, showing how the resampling process operates in theory. We extend this to a more general GridWorld-style Q-learning task which diverges under vanilla TD, but is consistently solved by POP-Q-learning.

A key strength of POP-Q-learning is that is achieves all this with a per-loop compute and memory overhead of the same order as Q-learning methods, and can be implemented and optimized in the same loop as any TD or Q-learning method. In this sense, it offers a cheap mechanism to stabilize off-policy TD, particularly in the context of offline RL.

A possible future expansion of this project is to integrate this with an existing offline RL method such as Conservative Q-Learning (CQL) and examine whether this improves performance. We propose CQL specifically because it constrains actions to remain within the support of the data, but does not explicitly constrain the distribution of states to minimize distribution shift. POP methods require adequate support (which CQL provides), and in turn are able to minimize distribution shift. This suggests that the two algorithms may have some symbiotic relationship.

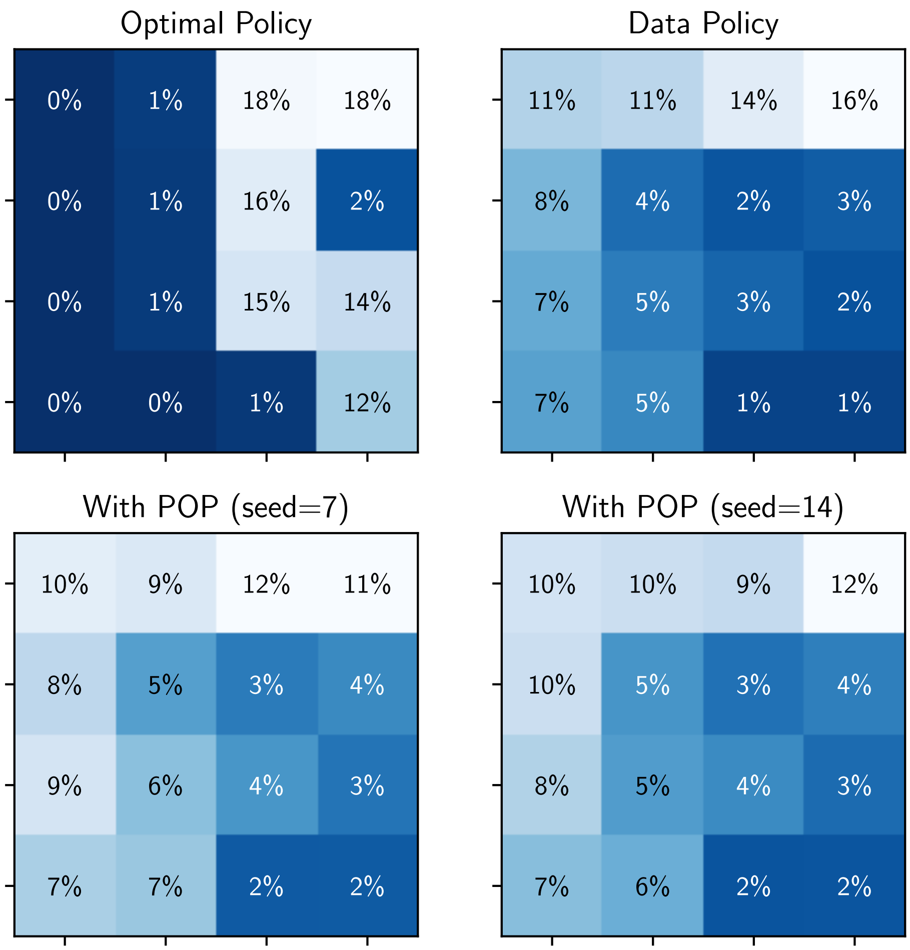


Figure 3.4: Rates at which states are visited in GridWorld. On the top row, we show how the optimal policy (left) is very far from the data policy (right). On the bottom row, we show the effective distribution after POP-Q resamples the data. The effective distribution is very close to the data distribution, despite the tremendous improvement in error.

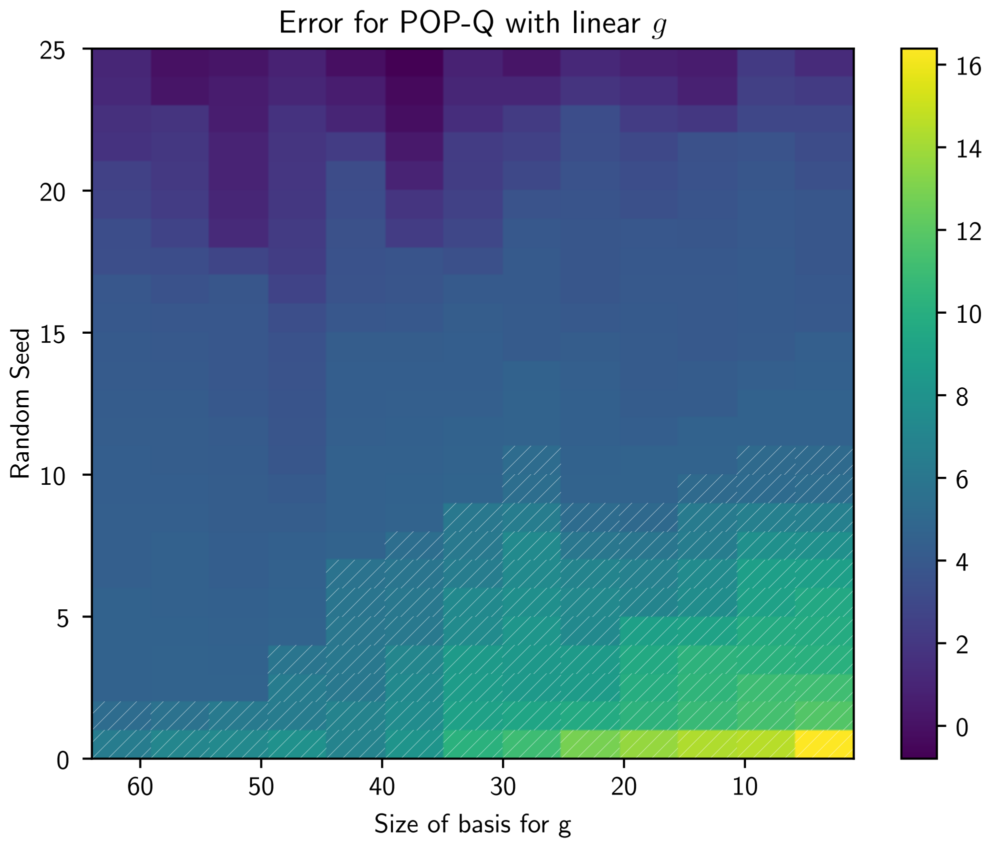


Figure 3.5: Error for POP-Q with linear g functions. Each row corresponds to one starting basis, and each column corresponds to a basis size l as it is reduced from 64 to 1. The hatched cells correspond to combinations of seeds and bases in which POP-Q performs worse than vanilla Q-learning. Under linear approximation POP-Q greatly improves performance over vanilla TD.

Conclusion

We have examined two notions of stability with a subtle relationship: that of learned dynamics models, and the training of reinforcement learning algorithms. In doing so, we have introduced new techniques in both areas, as well as filled in a gap in the literature on the unsuitability of regularization to solve instability in RL.

One key gap in the RL literature that we hope to address in the future is how we should regularize deep RL in a principled manner. While our prior work shows that simple ℓ_2 regularization can cause divergence, the literature is ripe for either adaptive regularization schemes that can detect and avoid pathological behavior, or for novel non-convex regularizations that fail similarly.

Separately, there remains a large gap in a key area within offline RL in dealing with the distributional shift problem. While there have been many recent advances in the field, these advances have been largely incremental, and the field remains ripe for a novel perspective that can address this. We propose that POP-TD is that novel perspective. Unlike the existing literature, the key insight that POP-TD brings to the field is that we can resample to "safe" off-policy distributions that are close to the data distribution, instead of the on-policy distribution which may be arbitrarily far. With these novel POP techniques, we hope to allow offline RL to resample the data as little as possible to avoid instability from large resampling coefficients, and learn to generalize from a set of diverse and possibly even adversarial experts that complete tasks in mutually incompatible ways.

Notation and Definitions

Standard notation for RL concepts through this thesis.

Symbol	Description
$n \in \mathbb{Z}^+$	Number of states.
$k \in \mathbb{Z}^+$	Number of features in the value basis.
$\pi \in \mathbb{R}^n$	on-policy distribution.
$\mu \in \mathbb{R}^n$	sampling distribution, may be on- or off-policy.
$\Phi \in \mathbb{R}^{[n \times k]}$	Feature basis for the value function
$\hat{w} \in \mathbb{R}^{[k \times 1]}$	Linear weights for value function, fit using least-squares regression of V on Φ.
$w^*(\eta) \in \mathbb{R}^{[k \times 1]}$	Linear weights for value function, learned using TD.
$\Phi w^*(\eta) \in \mathbb{R}^{[n \times 1]}$	Learned value function
$V \in \mathbb{R}^{[n \times 1]}$	True value function
$\|V\| \in \mathbb{R}$	Error from guessing zeros, equivalent to the threshold for a vacuous example
$\|x\| \in \mathbb{R}_0^+$	ℓ_2-norm of vector or matrix x, equal to $\sqrt{x^\top x}$
$\|x\|_D \in \mathbb{R}_0^+$	ℓ_2-norm of vector or matrix x under D, equal to $\sqrt{x^\top D x}$

Regularization

Symbol	Description
$\eta \in \mathbb{R}_0^+$	ℓ_2 regularization parameter
$h \in [0, 1]$	distribution parameter used to express a family of possible sampling distributions.
$\eta_m \in \mathbb{R}_0^+$	ℓ_2 regularization parameter for emphasis model in COF-PAC (the Emphatic algorithm we analyze)
$\eta_v \in \mathbb{R}_0^+$	ℓ_2 regularization parameter for value model in COF-PAC (the Emphatic algorithm we analyze)
$\upsilon : \mathbb{R}^+ \to \mathbb{R}^n$	apparent distribution induced by η-regularizing the emphatic correction of off-policy μ to on-policy π

Projected Off-Policy

Symbol	Description
$m \in \mathbb{Z}^+$	Number of features in the g-basis (for POP methods).
$l \in \mathbb{Z}^+$	Rank of A and B two-timescales parameters (for POP methods).
$g : \mathcal{S} \to \mathbb{R}$	dual objective component, learned opposite A and B in POP methods.
$e^{g(s)} \in \mathbb{R}^+$	The resampling coefficient for TD updates from state s
$\Phi_g \in \mathbb{R}^{[n \times m]}$	Feature basis for the learned linear g function
$w_g \in \mathbb{R}^{[m \times 1]}$	Linear weights for learned g function
$A, B \in \mathbb{R}^{[k \times l]}$	Two-timescales parameters learned alongside g in POP methods, where $l << k$.
$\Phi_g w_g \in \mathbb{R}^{[n \times 1]}$	Learned g function

Bibliography

[1] Brandon Amos, Lei Xu, and J Zico Kolter. "Input convex neural networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR.org. 2017, pp. 146–155.

[2] Caroline Blocher, Matteo Saveriano, and Dongheui Lee. "Learning stable dynamical systems using contraction theory". In: *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE. 2017, pp. 124–129.

[3] Byron Boots, Geoffrey J Gordon, and Sajid M Siddiqi. "A constraint generation approach to learning stable linear dynamical systems". In: *Advances in neural information processing systems*. 2008, pp. 1329–1336.

[4] Samuel Burer and Renato DC Monteiro. "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization". In: *Mathematical Programming* 95.2 (2003), pp. 329–357.

[5] Yize Chen, Yuanyuan Shi, and Baosen Zhang. "Optimal Control Via Neural Networks: A Convex Approach". In: *arXiv preprint arXiv:1805.11835* (2018).

[6] Yinlam Chow et al. "A lyapunov-based approach to safe reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2018, pp. 8092–8101.

[7] Jonas Degrave et al. "Magnetic control of tokamak plasmas through deep reinforcement learning". In: *Nature* 602.7897 (2022), pp. 414–419.

[8] Raghuram Bharadwaj Diddigi, Chandramouli Kamanchi, and Shalabh Bhatnagar. "A convergent off-policy temporal difference algorithm". In: *arXiv preprint arXiv:1911.05697* (2019).

[9] Simon S Du et al. “Stochastic variance reduction methods for policy evaluation”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1049–1058.

[10] William Fedus et al. “Revisiting fundamentals of experience replay”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 3061–3071.

[11] Scott Fujimoto, David Meger, and Doina Precup. “Off-policy deep reinforcement learning without exploration”. In: *International conference on machine learning*. PMLR. 2019, pp. 2052–2062.

[12] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. “Improving PILCO with Bayesian neural network dynamics models”. In: *Data-Efficient Machine Learning workshop, ICML*. Vol. 4. 2016.

[13] Carles Gelada and Marc G Bellemare. “Off-policy deep reinforcement learning by bootstrapping the covariate shift”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 3647–3655.

[14] Shixiang Gu et al. “Continuous deep q-learning with model-based acceleration”. In: *International Conference on Machine Learning*. 2016, pp. 2829–2838.

[15] Tuomas Haarnoja et al. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.

[16] Assaf Hallak and Shie Mannor. “Consistent on-line off-policy evaluation”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1372–1383.

[17] Hado van Hasselt et al. “Expected eligibility traces”. In: *arXiv preprint arXiv:2007.01839* (2021).

[18] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[19] Ray Jiang et al. “Learning Expected Emphatic Traces for Deep RL”. In: *arXiv preprint arXiv:2107.05405* (2021).

[20] Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems*. Vol. 3. Prentice hall Upper Saddle River, NJ, 2002.

[21] S Mohammad Khansari-Zadeh and Aude Billard. “Learning stable nonlinear dynamical systems with gaussian mixture models”. In: *IEEE Transactions on Robotics* 27.5 (2011), pp. 943–957.

[22] Rahul Kidambi et al. “Morel: Model-based offline reinforcement learning”. In: *Advances in neural information processing systems* 33 (2020), pp. 21810–21823.

[23] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[24] J Kolter. “The fixed points of off-policy TD”. In: *Advances in Neural Information Processing Systems* 24 (2011), pp. 2169–2177.

[25] Aviral Kumar, Abhishek Gupta, and Sergey Levine. “Discor: Corrective feedback in reinforcement learning via distribution correction”. In: *arXiv preprint arXiv:2003.07305* (2020).

[26] Aviral Kumar et al. “Conservative Q-Learning for Offline Reinforcement Learning”. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle et al. 2020. URL: `https://proceedings.neurips.cc/paper/2020/hash/0d2b2061826a5df3221116a5085a6052-Abstract.html`.

[27] Aviral Kumar et al. “DR3: Value-Based Deep Reinforcement Learning Requires Explicit Regularization”. In: *International Conference on Learning Representations*. 2022. URL: `https://openreview.net/forum?id=POvMvLi91f`.

[28] Aviral Kumar et al. “Stabilizing off-policy q-learning via bootstrapping error reduction”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[29] Joseph La Salle and Solomon Lefschetz. *Stability by Liapunov’s Direct Method with Applications by Joseph L Salle and Solomon Lefschetz*. Vol. 4. Elsevier, 2012.

[30] Sergey Levine et al. “Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems”. In: *CoRR* abs/2005.01643 (2020). arXiv: `2005.01643`. URL: `https://arxiv.org/abs/2005.01643`.

[31] Sergey Levine et al. "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems". In: *CoRR* abs/2005.01643 (2020). arXiv: 2005.01643. URL: https://arxiv.org/abs/2005.01643.

[32] Qiang Liu et al. "Breaking the curse of horizon: Infinite-horizon off-policy estimation". In: *Advances in Neural Information Processing Systems* 31 (2018).

[33] Sridhar Mahadevan et al. "Proximal reinforcement learning: A new theory of sequential decision making in primal-dual spaces". In: *arXiv preprint arXiv:1405.6757* (2014).

[34] Gaurav Manek and J Zico Kolter. "Learning stable deep dynamics models". In: *Advances in neural information processing systems* 32 (2019).

[35] Gaurav Manek and J Zico Kolter. "The Pitfalls of Regularization in Off-Policy TD Learning". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022. URL: https://openreview.net/forum?id=vK53GLZJes8.

[36] Gaurav Manek, Melrose Roderick, and J Zico Kolter. "Projected Off-Policy TD Learning Stabilize Offline Reinforcement Learning". In: Jan. 2023.

[37] Nikhil Mishra, Pieter Abbeel, and Igor Mordatch. "Prediction and control with temporal segment models". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 2459–2468.

[38] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 00280836. URL: http://dx.doi.org/10.1038/nature14236.

[39] Ofir Nachum et al. "Algaedice: Policy gradient from arbitrary experience". In: *arXiv preprint arXiv:1912.02074* (2019).

[40] Ofir Nachum et al. "Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections". In: *Advances in Neural Information Processing Systems* 32 (2019).

[41] Anusha Nagabandi et al. "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 7559–7566.

[42] Antonis Papachristodoulou and Stephen Prajna. "On the construction of Lyapunov functions using the sum of squares decomposition". In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.* Vol. 3. IEEE. 2002, pp. 3482–3487.

[43] Pablo A Parrilo. "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization". PhD thesis. California Institute of Technology, 2000.

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

[45] Spencer M Richards, Felix Berkenkamp, and Andreas Krause. "The lyapunov neural network: Adaptive stability certification for safe learning of dynamic systems". In: *arXiv preprint arXiv:1808.00924* (2018).

[46] Arno Schödl et al. "Video textures". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques.* ACM Press/Addison-Wesley Publishing Co. 2000, pp. 489–498.

[47] Laixi Shi et al. "Pessimistic q-learning for offline reinforcement learning: Towards optimal sample complexity". In: *arXiv preprint arXiv:2202.13890* (2022).

[48] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* Second Edition. MIT press, 2020.

[49] Richard S Sutton, A Rupam Mahmood, and Martha White. "An emphatic approach to the problem of off-policy temporal-difference learning". In: *The Journal of Machine Learning Research* 17 (2016), pp. 2603–2631.

[50] Richard S Sutton et al. "Fast gradient-descent methods for temporal-difference learning with linear function approximation". In: *Proceedings of the 26th Annual International Conference on Machine Learning.* 2009, pp. 993–1000.

[51] Andrew J Taylor et al. "Episodic Learning with Control Lyapunov Functions for Uncertain Robotic Systems". In: *arXiv preprint arXiv:1903.01577* (2019).

[52] Russ Tedrake. *Underactuated Robotics. Algorithms for Walking, Running, Swimming, Flying, and Manipulation.* 2023. URL: https://underactuated.csail.mit.edu.

[53] Andrey Nikolayevich Tikhonov. "On the stability of inverse problems". In: *Dokl. Akad. Nauk SSSR*. Vol. 39. 1943, pp. 195–198.

[54] JN Tsitsiklis and B Van Roy. "An analysis of temporal-difference learning with function approximation". In: *Rep. LIDS-P-2322). Lab. Inf. Decis. Syst. Massachusetts Inst. Technol. Tech. Rep* (1996).

[55] Jonas Umlauft and Sandra Hirche. "Learning stable stochastic nonlinear dynamical systems". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3502–3510.

[56] Jake VanderPlas. *Triple Pendulum CHAOS!* `http://jakevdp.github.io/blog/2017/03/08/triple-pendulum-chaos/`. Mar. 2017.

[57] Andrew J Wagenmaker et al. "First-Order Regret in Reinforcement Learning with Linear Function Approximation: A Robust Estimation Approach". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, July 2022, pp. 22384–22429. URL: `https://proceedings.mlr.press/v162/wagenmaker22a.html`.

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

[59] Ronald J Williams and Leemon C Baird III. *Analysis of some incremental variants of policy iteration: First steps toward understanding actor-critic learning systems*. Tech. rep. Citeseer, 1993.

[60] Yifan Wu, George Tucker, and Ofir Nachum. "Behavior regularized offline reinforcement learning". In: *arXiv preprint arXiv:1911.11361* (2019).

[61] Huizhen Yu. "On convergence of some gradient-based temporal-differences algorithms for off-policy learning". In: *arXiv preprint arXiv:1712.09652* (2017).

[62] Tianhe Yu et al. "Mopo: Model-based offline policy optimization". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 14129–14142.

[63] Shangtong Zhang, Hengshuai Yao, and Shimon Whiteson. "Breaking the Deadly Triad with a Target Network". In: *CoRR* abs/2101.08862 (2021). arXiv: `2101.08862`. URL: `https://arxiv.org/abs/2101.08862`.

[64] Shangtong Zhang et al. "Provably convergent two-timescale off-policy actor-critic with function approximation". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11204–11213.