Discount Factor as a Regularizer in Reinforcement Learning

Ron Amit [1] **Ron Meir** [1] **Kamil Ciosek** [2]

Abstract

Specifying a Reinforcement Learning (RL) task involves choosing a suitable planning horizon, which is typically modeled by a discount factor. It is known that applying RL algorithms with a lower discount factor can act as a regularizer, improving performance in the limited data regime. Yet the exact nature of this regularizer has not been investigated. In this work, we fill in this gap. For several Temporal-Difference (TD) learning methods, we show an explicit equivalence between using a reduced discount factor and adding an explicit regularization term to the algorithm's loss. Motivated by the equivalence, we empirically study this technique compared to standard L_2 regularization by extensive experiments in discrete and continuous domains, using tabular and functional representations. Our experiments suggest the regularization effectiveness is strongly related to properties of the available data, such as size, distribution, and mixing rate.

1. Introduction

The ability to perform well in new and unfamiliar situations following a limited learning experience is a hallmark of human intelligence. Similarly, the generalization ability of Reinforcement Learning (RL) algorithms is often measured by expected performance achieved by the agent in a Markov Decision Process (MDP) after being exposed to a limited amount of training data. Developing RL agents that generalize well is a longstanding challenge (Boyan & Moore, 1995; Sutton, 1996) that has recently been gaining more attention (Cobbe et al., 2018; Zhang et al., 2018b;a; Wang et al., 2019; Zhao et al., 2019). In particular, generalization is critical for successfully deploying RL agents that were trained in a simulator in complex real-world scenarios that contain elements not seen in the simulation.

There are several known approaches for improving generalization in RL. Selecting an appropriate function approximation model is one way to facilitate generalization across states and actions (Boyan & Moore, 1995). Regularization methods can further improve the generalization capacity. For example, it is very common to perform regularization in policy space by encouraging policies with high entropy (Williams, 1992; Mnih et al., 2016; Ahmed et al., 2019; Vieillard et al., 2020). Our focus is instead on policy evaluation. Traditionally, there have been two common approaches to such regularization. First, one can use traditional regularization methods from supervised learning to estimate the value function. Most commonly, this means adding an L_2 or L_1 penalty on the parameters of the value function (critic) (Kolter & Ng, 2009; Liu et al., 2012; Dann et al., 2014; Lillicrap et al., 2015; Cobbe et al., 2018; Liu et al., 2019). Second, one can apply indirect regularization by running the learning algorithm with a discount factor lower than specified by the task. We refer to this method as *discount regularization.* By focusing learning on short-term gains, this approach may improve generalization by reducing variance (Petrik & Scherrer, 2009; Jiang et al., 2015b;a; François-Lavet et al., 2019; van Seijen et al., 2019). This leads to the question:

What are the factors that influence the effectiveness of discount regularization?

This paper contributes to answering this question in three ways. First, for a few variants of TD learning, we show an equivalence between using a reduced discount and *activation regularization*, a technique used to train Recurrent Neural Networks (RNNs) (Merity et al., 2017; 2018; Herold et al., 2018).

Second, we empirically investigate the effectiveness of discount regularization in both tabular MDPs and large scale continuous control benchmarks. We show the benefit of discount regularization is strongly linked to the number of samples, uniformity of the state visitation and mixing rate of the data collection. Generally, discount regularization

[1]The Viterbi Faculty of Electrical Engineering, Technion - Israel Institute of Technology, Haifa, Israel [2]Microsoft Research, Cambridge, UK. Correspondence to: Ron Amit <ronamit@campus.technion.ac.il>, Ron Meir <rmeir@ee.technion.ac.il>, Kamil Ciosek <Kamil.Ciosek@microsoft.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

is more effective when data is limited, data distribution is highly uniform, and the mixing rate is low. In general, we fond discount regularization and L_2 regularization have similar performance in tabular settings, but vary in some function approximation settings.

Section 2 provides background on TD learning. In Section 3.1 we formalize the equivalence between using an artificially lowered discount and activation regularization. In Sections 4.1 and 4.2 we investigate our predictions empirically in tabular and deep RL benchmarks respectively. Section 5 discusses related work.

2. Background

2.1. Problem Setting

An MDP (Bellman, 1957) is defined as a tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, P, R, \mu)$, where $\mathcal{S}$ is the state set, $\mathcal{A}$ is the action set, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{M}(\mathcal{S})$ is the transition probability function, $\mathcal{M}(\mathcal{S})$ is the set of distributions over $\mathcal{S}$, $R : \mathcal{S} \times \mathcal{A} \to \mathcal{M}([0, R_{\max}])$ is the reward distribution function, $\mathcal{M}([0, R_{\max}])$ is the set of distributions supported on $[0, R_{\max}]$ and $\mu \in \mathcal{M}(\mathcal{S})$ is the initial state distribution. A Markovian stationary policy is defined by a mapping $\pi : \mathcal{S} \to \mathcal{M}(\mathcal{A})$. At each time-step t the agent draws an action a_t from $\pi(s_t)$ where s_t is the current state. The agent then receives a random reward $r_t \sim R(s_t, a_t)$ and transitions to the next state s_{t+1} drawn from $P(s_t, a_t)$. This process produces a (possibly infinite) trajectory $\tau := (s_0, a_0, r_0, s_1, a_1, r_1, ...)$. Given a discount factor $\gamma \in [0, 1]$ the value function at state s is defined by the expected discounted return $V^\pi_\gamma(s) := \mathbb{E}_{\tau:s_0=s}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s]$. Similarly we define the Q-function given state s and action a as $Q^\pi_\gamma(s, a) := \mathbb{E}_{\tau:s_0=s,a_0=a}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a]$.

In our setting, the agent is allowed to observe a limited number of samples of trajectories generated from $\mathcal{M}$. We define a sample as a single transition (s, a, r, s'), where s is the current state, a is the action taken, r is the immediate reward , and s' is the next state. We investigate two types of goals: *policy evaluation* and *control*. In policy evaluation the agent is given a fixed policy π and aims to estimate $V^\pi_{\gamma_e}(s)$ where $\gamma_e \in [0, 1]$ is the *evaluation discount factor*. In the control setting the agent aims to find a policy π that maximizes the expected return $\mathbb{E}_{\tau:\pi}[\sum_{t=0}^{\infty} \gamma_e^t r_t]$. In this paper we investigate control algorithms that include policy evaluation as one constituent component.

We consider policy evaluation with function approximation, where the estimated value function is chosen from a parametric family $\left\{\hat{V}_\theta : \mathcal{S} \to \mathbb{R} | \theta \in \mathbb{R}^d\right\}$. We assume the functions in this family are differentiable w.r.t. θ. The tabular setting can be considered as a special case with $\hat{V}_\theta(s) := \theta_s, \theta \in \mathbb{R}^{|\mathcal{S}|}$.

2.2. Temporal-Difference Learning

The Temporal-Difference (TD) learning algorithm family (Sutton, 1988) is used for efficient policy evaluation. While our insights apply to a wide range of TD methods, we focus our discussion on TD(0) as a representative algorithm. We address the m-step variant and the SARSA algorithm in Appendices A.2 and A.3. We will consider a batch setting, in which the task is to estimate the value function $V^\pi_{\gamma_e}(s)$ of a known policy π given samples from trajectories generated by interaction of π with the MDP $\mathcal{M}$. We assume the finite data setting, i.e, we are given a data set D of n samples. Since we are interested in effects of finite sample size and not a finite number of iterations, we choose to focus on the batch rather than an online setting. In the batch setting, we can reuse each sample in the data set for many iterations.

Algorithm 1 is a generic form of a regularized batch TD(0) algorithm. In the special case of the standard non-regularized TD(0), there is no added regularization term ($\Psi \equiv 0$), there is no reward scaling $\xi = 1$, and the discount factor used is the one desired in the problem definition ($\gamma = \gamma_e$). The algorithm is initialized at some initial parameters θ_0 and takes steps aiming to minimize $\mathbb{E}_{(s,a,r,s')\sim D}\left\{\left(r + \gamma \hat{V}_{\tilde{\theta}}(s') - \hat{V}_\theta(s)\right)^2 + \Psi\right\}$, where the expectation is w.r.t. a the empirical distribution over samples D. Similarly to Stochastic Gradient Descent (SGD), in each iteration only one transition (s, a, r, s') is sampled from D to approximate the full gradient. For stability considerations, instead of the standard gradient, the algorithm computes a *'semi-gradient'* (Sutton & Barto, 2018), i.e. the next state value estimate, $\hat{V}_{\tilde{\theta}}(s')$, is fixed. The learning rate $\alpha_i \in \mathbb{R}^+$ is usually set to be monotonically decaying at rate $O(1/i)$ in table-lookup settings and scaled automatically (Kingma & Ba, 2015) in deep learning settings.

Algorithm 1 Generic Regularized Batch TD(0)

Hyper-parameters: $\gamma \in [0, \gamma_e]$, $\xi \in \mathbb{R}^+$ (global reward scaling), Ψ (regularization function)
Input: D
for $i = 0, 1, ..., N_{\text{iter}} - 1$ **do**
  Get uniformly random (s, a, r, s') from D
  $\theta_{i+1} := \theta_i + \alpha_i\left(\xi r + \gamma \hat{V}_{\theta_i}(s') - \hat{V}_{\theta_i}(s)\right)\nabla \hat{V}_{\theta_i}(s) - \alpha_i \nabla(\Psi)$.
end for

We are interested in the result in the limit of an infinite number of iterations $N_{\text{iter}} \to \infty$ for which all samples from D are used infinitely often. Note that since we are dealing with finite data, convergence to the true value is not guaranteed even for $\gamma := \gamma_e$. We refer to the discount factor $\gamma \in (0, 1)$ used by the algorithm as the *guidance discount factor* (Jiang et al., 2015b). In this paper we study the regularizing effect

of using γ lower than the evaluation discount factor γ_e and compare it with other regularization methods.

Q-function evaluation. In many cases (e.g., control) we are interested in estimating the action-value Q^π function rather than V^π. The naive variant of TD(0) for estimating the Q-function is the SARSA(0) algorithm (Rummery & Niranjan, 1994). In Appendix A.2, we also discuss a variant called Expected SARSA(0) (Sutton et al., 1998) which utilizes knowledge of π to perform lower variance updates (Van Seijen et al., 2009).

Policy iteration. In our work, we investigate control algorithms that fit the policy iteration framework, i.e, algorithms that alternate between policy evaluation and policy improvement. Specifically, we investigate algorithms that use TD-style policy evaluation. Many control RL algorithms fit this framework, including modern actor-critic methods such as DPG (Silver et al., 2014), SAC (Haarnoja et al., 2018), DDPG (Lillicrap et al., 2015), and Twin Delayed DDPG (TD3) (Fujimoto et al., 2018), which is investigated in the experiments section.

3. Discount Regularization in TD Learning

3.1. Equivalence of Reduced Discount Factor and Activation Regularization

In this section, we formulate the equivalence between TD learning with a reduced discount and TD learning with a high discount with an added regularization term. The equivalence will provide insights about the effectiveness of discount regularization in various settings.

For simplicity of presentation, we first show that TD(0) with guidance discount factor $\gamma < \gamma_e$ is equivalent to an added activation regularization term to the standard γ_e-discounted update. Analogous results can obtained for SARSA (Appendix A.2), m-step TD (Appendix A.3) and LSTD (Appendix A.4). The proof is in Appendix A.1.

Proposition 1. *Let $\theta_1, \theta_2, \ldots$ be the parameters produced by Algorithm 1 using a discount factor $\gamma < \gamma_e$, with $\xi = 1, \Psi \equiv 0$, initial parameters θ_0 and learning rate α_i. The algorithm, produces the same sequence of parameters $\theta_1, \theta_2 \ldots$ if it is run with the discount factor $\gamma = \gamma_e$, but with added regularization function $\Psi(s, \theta) := \lambda\left(\hat{V}_\theta(s)\right)^2$, $\lambda := \frac{\gamma_e - \gamma}{2\gamma}$, reward scaling $\xi := \frac{\gamma_e}{\gamma}$, learning rate $\alpha'_i := \frac{\gamma}{\gamma_e}\alpha_i$ and the same initial parameters θ_0.*

Proposition 1 implies that running TD(0) with a reduced discount factor is equivalent to minimizing the objective $\mathbb{E}_{(s,a,r,s')\sim D}\left\{\left(\xi r + \gamma_e \hat{V}_{\tilde{\theta}}(s') - \hat{V}_\theta(s)\right)^2 + \lambda\left(\hat{V}_\theta(s)\right)^2\right\}$. We refer to the added regularization term as *activation regularization*[1]. In TD(0), this term is the mean value of the square of the learned value function over the distribution of observed states $\lambda\mathbb{E}_s\left(\hat{V}_\theta(s)\right)^2$. In the SARSA algorithm we have a similar term $\lambda\mathbb{E}_{(s,a)}\left(\hat{Q}_\theta(s,a)\right)^2$ (see Appendix A.2). This term penalizes large value estimates and therefore encourages consistent value estimates across state-action pairs, which may encourage generalization by reducing the effect of spurious approximation errors. Reducing γ increases the factor of the equivalent regularization term $\lambda := \frac{\gamma_e - \gamma}{2\gamma}$.

We can get a more explicit form for the activation regularization when using a tabular function or a linear approximation with orthogonal features, where the activation regularization term is equal to a weighted L_2 norm on the parameters. Define $\hat{V}_\theta(s) := \phi(s)^\top\theta$ for some fixed feature mapping ϕ and some weight vector $\theta \in \mathbb{R}^k$. Assume orthogonal features, i.e, that we have[2] $\mathbb{E}_s\left[\phi(s)\phi(s)^\top\right] = \Lambda$ for some diagonal matrix Λ. This assumption holds for the tabular case for which $\phi(s) = e_s$ where e_s is the standard basis of $\mathbb{R}^{|\mathcal{S}|}$. The activation regularization term can be written as

$$\begin{aligned}\lambda\mathbb{E}_s\left(\hat{V}_\theta(s)\right)^2 &= \lambda\mathbb{E}_s\left[(\theta^\top\phi(s)\phi(s)^\top\theta)\right] \\ &= \lambda\theta^\top\Lambda\theta = \lambda\|\theta\|^2_\Lambda.\end{aligned} \tag{1}$$

If, in addition, the features are also *orthonormal*, i.e, $\Lambda = E_s\left[\phi(s)\phi(s)^\top\right] = \mathbb{I}_{k\times k}$ then the activation regularization term becomes equivalent to the an L_2 regularization term $\|\theta\|_2^2$. For example, this case applies for tabular representation when the data distribution is uniform across states. Note that even in this case, if we want discount regularization to be equivalent to L_2 regularized algorithm with $\gamma := \gamma_e$, Proposition 1 claims that we should adjust the reward scaling and learning rate: $\xi := \frac{\gamma_e}{\gamma}, \alpha'_i := \frac{\gamma}{\gamma_e}\alpha_i$ (i.e, the inverse transformation to the one described in the proposition).

Proposition 1 showed that a reduced discount is equivalent to adding a activation regularization term $\lambda\mathbb{E}_{s\sim D}\left(\hat{V}_\theta(s)\right)^2$ to the learning objective. Notice that this term is sensitive to the distribution over the observed states. For example, in the tabular case, $\hat{V}_\theta(s) := \theta_s$, the activation regularization term is simplified to $\lambda\mathbb{E}_{s\sim D}\theta_s^2$. This form demonstrates that states that are visited less often are less regularized, i.e, the regularization factors for these states are lower. If a state is not visited at all, the value estimation for this state is not regularized at all.

[1] This naming relates to activation regularization in RNNs, which refers to L_2 penalty on the RNN activations, rather than on the weights of the network in standard L_2 regularization (Merity et al., 2017; 2018)

[2] The expectation is w.r.t a uniform distribution over the samples $(s, a, r, s') \in D$.

This phenomenon raises a concern that activation regularization (or equivalently small discount) may be less helpful for generalization state visitation is farther from a uniform distribution. In Section 4.1 we will demonstrate empirically that discount regularization is indeed less beneficial when the data distribution is highly non-uniform.

4. Empirical Demonstrations

The goal of the of the experiments in this section is to investigate the following questions[3]. Can reducing the discount factor improve generalization performance with TD learning? How is the optimal discount factor related to data size? What is the effect of data uniformity and mixing rate? What is the benefit of discount regularization compared to L_2 regularization (in both tabular and function approximation settings)?

4.1. Tabular Experiments

We first investigate the effectiveness and discount regularization in various setting we conducted a simple GridWorld experiment.

In the GirdWorld environment the state space is a 4×4 grid, and the agent can move to along the gird. In each experiment, we randomly choose a 'goal state' to be assigned with a high reward mean. The other reward means and the transition probabilities are also generated randomly. The full details of the experiment appear in Appendix A.5.

We first experiment in a batch policy evaluation setting, for a fixed uniform policy. The evaluation metric we use is the L_2 distance of the estimated value from the true value $V^\pi_{\gamma_e}$, $\|\hat{V} - V^\pi_{\gamma_e}\|_2$, where $V^\pi_{\gamma_e}$ is evaluated with $\gamma_e = 0.99$. In our first set of experiments we generate the data by simulating trajectories starting at a random initial state and following a uniform policy for 50 time-steps. We varied the number of trajectories to change the sample size.

The results are summarized in Figure 3. Each plot shows the average loss across 1000 MDP instances and the 95% confidence intervals. In Figure 1(a), we clearly see that using a smaller discount factor $\gamma < \gamma_e$ can significantly improve performance when the available data set is small. This corresponds to our observation that a smaller discount is equivalent to a stronger activation regularization term. In Figure 1(b), we see the effect of L_2 regularization with no discount regularization ($\gamma = \gamma_e$). The results show L_2 regularization achieves similar performance gain as discount regularization. Figures 1(c) and 1(d) show the corresponding results with the LSTD algorithm. In contrast to TD(0), for LSTD we see that regularization can improve performance for all data set sizes, and the loss when not using regularization is higher.

In some case, the actual values of the estimates are less important than the relative rankings of the values of states. Therefore, we repeated the experiment with a loss function that compares state rankings (see Appendix A.6.1). The results show similar behaviour as with the L_2 loss.

We have seen that regularization is more helpful when the data size is limited. But there are other properties of the data that indicate that regularization may be more effective. Next, we will investigate the influence of the uniformity of the data distribution and of the mixing rate of the data generating process.

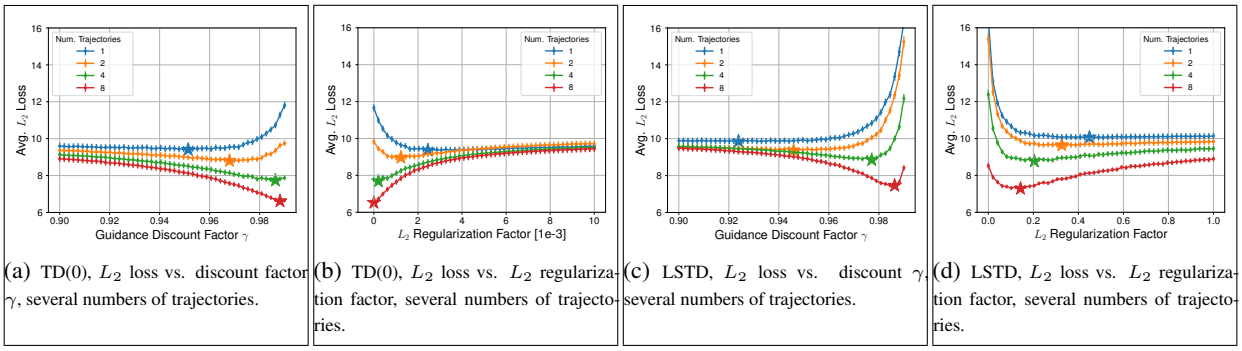
Influence of the uniformity of the data state distribution. We consider a batch setting, where the state-action tuples are drawn independently from fixed distributions (while the reward, next state, and next action are drawn according to the environment stochasticity and the evaluated policy). To measure the uniformity of the distributions, we evaluated the total variation distance from a uniform distribution. In each experiment repetition, we randomly generated distributions with various distances via rejection sampling. The data consists of 400 sampled tuples.

Figures 2(a) and 2(b) show the loss when using each of the regularization methods, for various distances from a uniform distribution, when using the LSTD algorithm. As seen in the figure, for data distributions close to uniform, the benefit of regularization is greater. In section 3.1 we predicted that discount regularisation will be more helpful for more uniform distributions. Interestingly, we find that the effectiveness of L_2 regularization is influenced in the same manner as discount regularization.

Mixing-time influence. Another interesting question regards the effect of the mixing-time on regularization effectiveness. In Markov chains, the mixing-time describes the typical convergence time of the state distribution to the stationary distribution. It can be computed using the inverse spectral gap of the transition probabilities matrix (Levin & Peres, 2017; Jerison, 2013).

To create trajectories with a specific mixing time we augmented the transition probabilities matrix to have the appropriate spectral gap for the specified mixing time (see full details in Appendix A.5.3). We study a batch policy evaluation setting, where the behavioral policy is uniform, and the data is collected from two trajectories of length 50. The value is estimated using LSTD. In each experiment repetition, we randomly create an MDP, derive the Markov process induced by a uniform policy, and apply the mixing time augmenting procedure. As seen in Figures 2(c)

[3]Code for all the experiments is available at: https://github.com/ron-amit/Discount_as_Regularizer.



(a) TD(0), L_2 loss vs. discount factor γ, several numbers of trajectories.

(b) TD(0), L_2 loss vs. L_2 regularization factor, several numbers of trajectories.

(c) LSTD, L_2 loss vs. discount γ, several numbers of trajectories.

(d) LSTD, L_2 loss vs. L_2 regularization factor, several numbers of trajectories.

Figure 1. **Tabular experiments - effect of dataset size.** Loss vs. regularization factor for different regularizers, and algorithms, averaged over 1000 MDP instances. In each figure, the curves correspond to different number of samples per episode. The star shapes mark the minimum of the curve. Error bars represent 95% confidence interval.

and 2(d), discount regularization and L_2 regularization are more effective in the slow mixing regime. Intuitively, in this regime, limited data is less representative of the whole state space, which leads to higher estimation variance and so more regularization is needed.

Policy optimization. Improving performance of policy evaluation with regularization can improve performance of policy-iteration based algorithms. To demonstrate this, we run 5 episodes of approximate policy iteration: *(i)* gather data by generating trajectories with 10 time-steps by rolling out ε-greedy policy with $\varepsilon = 0.1$, *(ii)* run policy evaluation with SARSA, and *(iii)* derive greedy policy w.r.t estimated value function. The evaluation metric, *optimality loss*, is the L_1 distance of the value of the learned policy V^π to the value of the optimal policy V^*, $\|V^\pi - V^*\|_1$, where the values are computed with the true model and $\gamma_e = 0.99$.

In Figures 3(a) and 3(b) we see the results for discount and L_2 regularization respectively. Both methods can achieve similar performance improvement. As in previous experiments, when less data is available, stronger regularization is needed. Note that while this experiment only tested one regularizer at a time, using a combination of both L_2 and discount regularization can considerably improve generalization, as seen in Figure 4.

4.2. Deep RL Experiments

In this section, we investigate whether a reduced discount (or equivalently activation regularization) will benefit generalization from a finite sample in a continuous control with function approximation setting. Our experiments use the Mujoco environment (Todorov et al., 2012). To test the ability to generalize from finite data, we limited the number of time-steps from the environment to 200,000 or less.

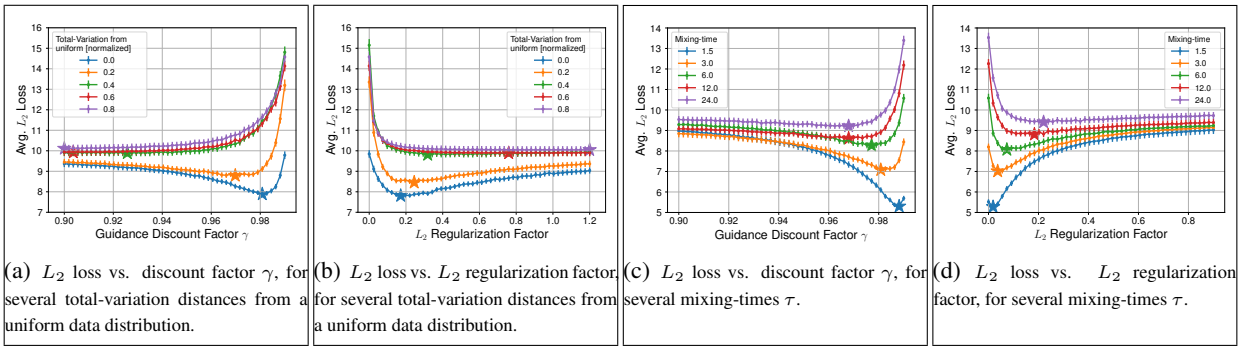
As a learning algorithm, we used the Twin Delayed DDPG (TD3) algorithm (Fujimoto et al., 2018), a recent actor-critic algorithm that achieves state-of-the-art performance in continuous control tasks. The policy evaluation stage of TD3 uses a variant of expected SARSA called target policy smoothing to estimate state-action values. Similar experiments with the DDPG algorithm (Lillicrap et al., 2015) are in Appendix A.8.

All hyper-parameters are identical to those suggested by (Fujimoto et al., 2018) except the following changes. We tested with several amounts of total time-steps to simulate a limited data setting. As in Fujimoto et al. (2018), The first 10^4 time steps are used only for exploration. Another change to improve learning stability is increasing the batch size from 100 to 256. See Appendix A.7 for the complete implementation details. We tested two regularization methods: *(i) discount regularization* - γ is varied and the L_2 factor is zero. *(ii) L_2 regularization* - the L_2 factor is varied and γ is fixed to high value of 0.999.

Since the focus of this paper is regularization of the value estimation phase, we tested L_2 regularization only for the critic network. As in common practice in deep learning, only the non-bias weight parameters are regularized and since they are less prone to over-fitting (Goodfellow et al., 2016).

For each tested hyper-parameter we repeated the experiments for 20 different initial random seeds. The averaging over a number of seeds allows for statistically significant results despite the high variance of the simulation environment (Henderson et al., 2018). In each repetition, the performance evaluation of the final policy is done by averaging the total undiscounted return (i.e, $\gamma_e = 1$) on 1000 new episodes.

The results appear in Figure 5. The results demonstrates that discount regularization can lead to significant performance gain. In the case of 200,000 time-steps, we can see that γ values of around 0.99 are optimal. For lower numbers of time-steps, lower discount factors are generally more favourable. For example, in the Ant-v2 experiment $\gamma = 0.8$



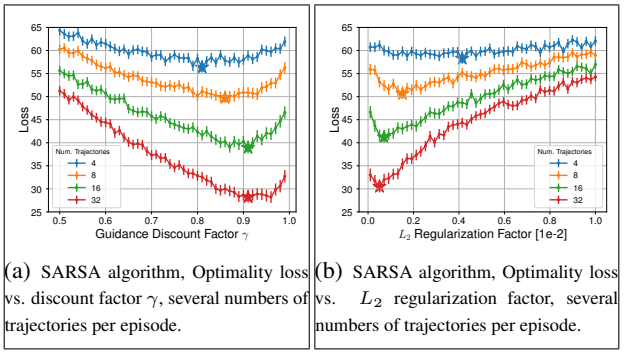
(a) L_2 loss vs. discount factor γ, for several total-variation distances from a uniform data distribution.

(b) L_2 loss vs. L_2 regularization factor, for several total-variation distances from a uniform data distribution.

(c) L_2 loss vs. discount factor γ, for several mixing-times τ.

(d) L_2 loss vs. L_2 regularization factor, for several mixing-times τ.

Figure 2. **Tabular Experiments - effect of data properties.** Loss vs. regularization factor for different regularizers, averaged over 1000 MDP instances. All results are with the LSTD algorithm. The star shapes mark the minimum of the curve. Error bars represent 95% confidence interval.



(a) SARSA algorithm, Optimality loss vs. discount factor γ, several numbers of trajectories per episode.

(b) SARSA algorithm, Optimality loss vs. L_2 regularization factor, several numbers of trajectories per episode.

Figure 3. **Tabular experiments - policy optimization.** Optimality loss vs. regularization factor for different regularizers, averaged over 1000 MDP instances. The number of trajectories per episode is 16. The star shapes mark the minimum of the curve. Error bars represent 95% confidence interval.

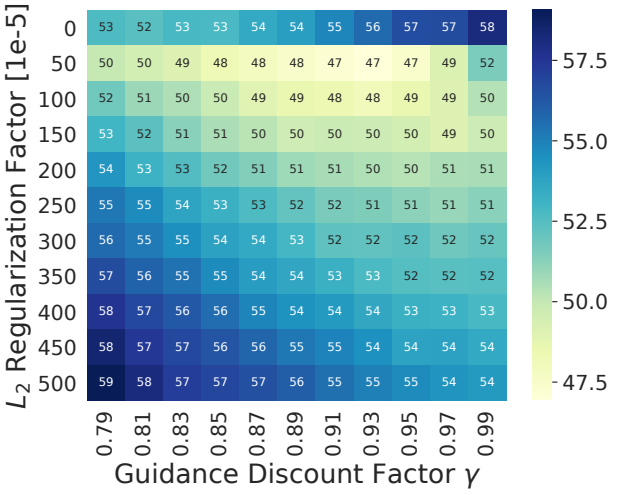


Figure 4. Optimality loss per guidance discount factor γ and L_2 factor. Results for 5 episodes of policy-iteration with 8 trajectories of length 10 per episode. The results are averaged across 1000 MDP instances from the environment. The 95% confidence interval is less than 2.5% relative to the mean.

is optimal for 100,000 time-steps (Fig. 5(g)).

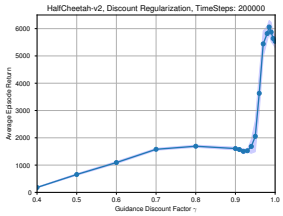
If we compare L_2 regularization to discount regularization, we see that sometimes it gives lower performance gain (e.g, Fig. 5(e) and 5(b)), but in other cases it gives a higher gain, especially for smaller amount of time-steps (e.g., Fig. 5(d) and 5(k)).

We note that there is a wide variability of behavior across the different Mujoco tasks (as has been observed also in previous work (Ahmed et al., 2019)). In practice, the discount factor should be chosen using a grid search for a specific environment and amount of available data. However, our work suggests a few helpful guidelines: if less data is available, lower discounts become more favourable, in scenarios with non-uniform data coverage, or a fast mixing time, lowering the discount is likely to be less helpful.

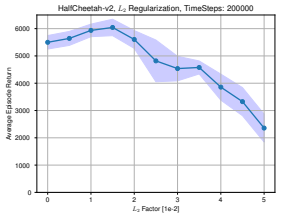
Note that the common practice in actor-critic algorithms for learning Mujoco environments is to regularize the policy evaluation by setting $\gamma = 0.99$ and L_2 factor of about 10^{-2} (e.g, Lillicrap et al. (2015)). Our results suggest that this hyper-parameter choice works well in some cases, but in other cases increasing the amount of regularization can significantly improve final performance.

5. Related Work

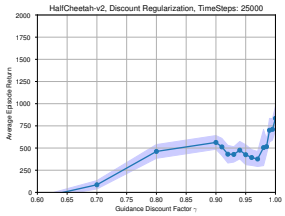
It is well-known that lower γ increases convergence rate in many RL algorithms (Bertsekas & Tsitsiklis, 1996), but several works showed that it can also improve final performance in the cae of limited data or approximation error. Petrik & Scherrer (2009) studied approximate dynamic-programming and showed that planning with a lower discount factor might be advised when the approximation error is large. Chen et al.



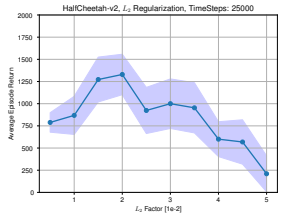
(a) HalfCheetah-v2, discount regularization, 2e5 steps



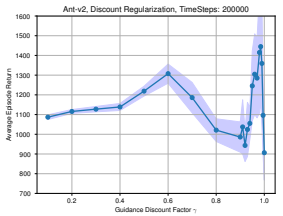
(b) HalfCheetah-v2, L_2 regularization, 2e5 steps



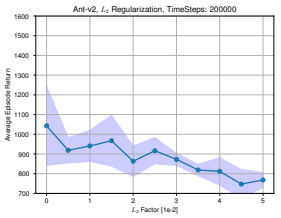
(c) HalfCheetah-v2, discount regularization, 25e3 steps



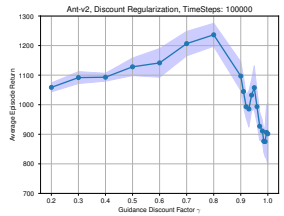
(d) HalfCheetah-v2, L_2 regularization, 25e3 steps



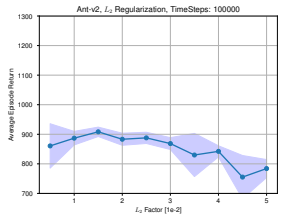
(e) Ant-v2, discount regularization, 2e5 steps



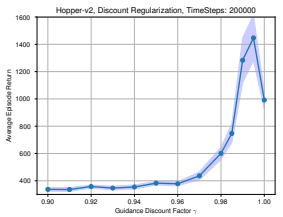
(f) Ant-v2, L_2 regularization, 2e5 steps



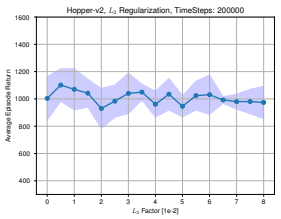
(g) Ant-v2, discount regularization, 1e5 steps



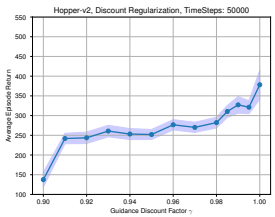
(h) Ant-v2, L_2 regularization, 1e5 steps



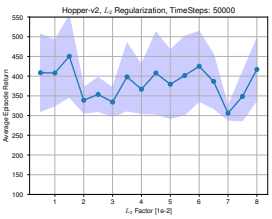
(i) Hopper-v2, discount regularization, 2e5 steps



(j) Hopper-v2, L_2 regularization, 2e5 steps



(k) Hopper-v2, discount regularization, 5e4 steps



(l) Hopper-v2, L_2 regularization, 5e4 steps

Figure 5. **Regularization in Mujoco experiments with limited data and TD3 algorithm.** Average total reward in evaluation episodes vs. regularization factor. Results are averaged over 20 simulations and 1000 evaluation episodes. Shaded area represent 95% confidence interval.

(2018) and François-Lavet et al. (2019) studied similar phenomena in POMDPs. Jiang et al. (2015b; 2016) studied a model-based RL setting and suggested that in the limited data regime, the performance of model-based RL can be improved by using a low discount factor in the planning phase. Our work identifies new elements that contribute to the effectiveness of discount regularization: uniformity and mixing rate.

In the planning setting, a classic result by Blackwell (1962) shows that for every finite MDP, there exists a discount factor γ^* such that planing with any greater discount factor ($\gamma \geq \gamma^*$) leads to an optimal policy in the average reward sense. (Kakade, 2001) showed that for faster mixing MDPs, lowered discount factors introduces less bias int the average reward sense. Our work shows that in the learning setting, lowered discounts can even allow better generalization in faster mixing scenarios.

The importance of regularization of generalization has also been demonstrated empirically with deep RL algorithms. Cobbe et al. (2018) suggested benchmarks for measuring generalization in deep RL and demonstrated that common regularization methods like L_2, can significantly improve generalization using the PPO algorithm (Schulman et al., 2017). Farebrother et al. (2018) showed regularization can improve generalization in Atari benchmarks when using the DQN algorithm (Mnih et al., 2015). Parisi et al. (2019) suggested a method for regularizing actor-critic algorithms by adding a TD error penalty in the actor's objective. Prokhorov & Wunsch (1997) demonstrated the benefit of discount regularization using a schedule for increasing γ as learning progresses. Similar scheduling is used in modern large scale RL applications (OpenAI, 2018). Xu et al. (2018) showed a gradient-based automatic hyper-parameter tuning method that achieved significant performance enhancement by tuning the discount. Sherstan et al. (2019) and Romoff et al. (2019) suggested methods for TD learning with a high discount via learning a sequence of value functions with lower discount factors. A recent line of works (Efroni et al., 2018; Tomar et al., 2019; Tessler & Mannor, 2020) proposes al-

gorithmic schemes for using a small discount factor that asymptotically converge to the solution of the problem with the original discount.

While the benefits of a low discount factor have been shown in some settings, in other settings it has been shown to have adverse effects. The work of van Seijen et al. (2019) analyze a family of small MDPs and show the existence of a sweet-spot in γ selection.

6. Conclusions

In this paper, we studied the regularization effect of using a low discount factor in RL algorithms. In summary, our work demonstrated empirically that discount regularization can significantly improve generalization performance when learning from limited data. In the tabular setting, we demonstrated that discount regularization is more effective for more uniform empirical state distribution or slower mixing rate. In our experiments, discount and L_2 regularization had similar performance gain in the tabular settings, but different gains in the deep RL settings.

Our work opens several directions for further research. *(i)* Can theoretical results explain the phenomena observed in our experiments? *(ii)* Can we explain the variation in performance between discount and L_2 regularization in the function approximation setting? *(iii)* Can we develop RL algorithms that utilize L_2 and discount regularization in an adaptive manner?

ACKNOWLEDGMENTS

We thank Yonatan Efroni, Tom Zahavy, Nadav Merlis, Chen Tessler, Nir Baram, Ester Dorfman, Asaf Cassel, Guy Tennenholtz, Baruch Epstein, Tom Jurgenson, Alekh Agarwal, Tom Minka, Katja Hofmann and the Game Intelligence team at Microsoft Research, for helpful discussions of this work, and the anonymous reviewers for their helpful comments. The work of RM is partially supported by grant 451/17 from the Israel Science Foundation, by the Ollendorff Center of the Viterbi Faculty of Electrical Engineering at the Technion, and by the Skillman chair in biomedical sciences.

References

Ahmed, Z., Le Roux, N., Norouzi, M., and Schuurmans, D. Understanding the impact of entropy on policy optimization. In *International Conference on Machine Learning*, pp. 151–160, 2019.

Bellman, R. A markovian decision process. *Journal of mathematics and mechanics*, pp. 679–684, 1957.

Bertsekas, D. P. and Tsitsiklis, J. N. *Neuro-dynamic programming*, volume 5. Athena Scientific Belmont, MA, 1996.

Blackwell, D. Discrete dynamic programming. *The Annals of Mathematical Statistics*, pp. 719–726, 1962.

Boyan, J. A. and Moore, A. W. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in neural information processing systems*, pp. 369–376, 1995.

Chen, Y.-C., Kochenderfer, M. J., and Spaan, M. T. Improving offline value-function approximations for pomdps by reducing discount factors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3531–3536. IEEE, 2018.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

Dann, C., Neumann, G., and Peters, J. Policy evaluation with temporal differences: A survey and comparison. *The Journal of Machine Learning Research*, 15(1):809–883, 2014.

Efroni, Y., Dalal, G., Scherrer, B., and Mannor, S. Beyond the one-step greedy approach in reinforcement learning. In *International Conference on Machine Learning*, pp. 1386–1395, 2018.

Farebrother, J., Machado, M. C., and Bowling, M. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

François-Lavet, V., Rabusseau, G., Pineau, J., Ernst, D., and Fonteneau, R. On overfitting and asymptotic bias in batch reinforcement learning with partial observability. *Journal of Artificial Intelligence Research*, 65:1–30, 2019.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, pp. 1582–1591, 2018.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT Press, 2016.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pp. 1856–1865, 2018.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Herold, C., Gao, Y., and Ney, H. Improving neural language models with weight norm initialization and regularization. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 93–100, 2018.

Jerison, D. General mixing time bounds for finite markov chains via the absolute spectral gap. *arXiv preprint arXiv:1310.8021*, 2013.

Jiang, N., Kulesza, A., and Singh, S. Abstraction selection in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 179–188, 2015a.

Jiang, N., Kulesza, A., Singh, S., and Lewis, R. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1181–1189. International Foundation for Autonomous Agents and Multiagent Systems, 2015b.

Jiang, N., Singh, S. P., and Tewari, A. On structural properties of MDPs that bound loss due to shallow planning. In *IJCAI*, pp. 1640–1647, 2016.

Kakade, S. Optimizing average reward using discounted rewards. In *International Conference on Computational Learning Theory*, pp. 605–615. Springer, 2001.

Kendall, M. G. Rank correlation methods. 1948.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Kolter, J. Z. and Ng, A. Y. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 521–528. ACM, 2009.

Lagoudakis, M. G. and Parr, R. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec): 1107–1149, 2003.

Levin, D. A. and Peres, Y. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Liu, B., Mahadevan, S., and Liu, J. Regularized off-policy td-learning. In *Advances in Neural Information Processing Systems*, pp. 836–844, 2012.

Liu, Z., Li, X., Kang, B., and Darrell, T. Regularization matters in policy optimization, 2019.

Merity, S., McCann, B., and Socher, R. Revisiting activation regularization for language rnns. *arXiv preprint arXiv:1708.01009*, 2017.

Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations (ICLR)*, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

Parisi, S., Tangkaratt, V., Peters, J., and Khan, M. E. Td-regularized actor-critic methods. *Machine Learning*, pp. 1–35, 2019.

Petrik, M. and Scherrer, B. Biasing approximate dynamic programming with a lower discount factor. In *Advances in neural information processing systems*, pp. 1265–1272, 2009.

Prokhorov, D. V. and Wunsch, D. C. Adaptive critic designs. *IEEE transactions on Neural Networks*, 8(5):997–1007, 1997.

Romoff, J., Henderson, P., Touati, A., Brunskill, E., Pineau, J., and Ollivier, Y. Separating value functions across timescales. In *International Conference on Machine Learning*, pp. 5468–5477, 2019.

Rummery, G. A. and Niranjan, M. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, England, 1994.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sherstan, C., Dohare, S., MacGlashan, J., Günther, J., and Pilarski, P. M. Gamma-nets: Generalizing value estimation over timescale. *arXiv preprint arXiv:1911.07794*, 2019.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, pp. 387–395, 2014.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Sutton, R. S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pp. 1038–1044, 1996.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Sutton, R. S., Barto, A. G., et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

Tessler, C. and Mannor, S. Maximizing the total reward via reward tweaking. *arXiv preprint arXiv:2002.03327*, 2020.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Tomar, M., Efroni, Y., and Ghavamzadeh, M. Multi-step greedy policies in model-free deep reinforcement learning. *arXiv preprint arXiv:1910.02919*, 2019.

Van Seijen, H., Van Hasselt, H., Whiteson, S., and Wiering, M. A theoretical and empirical analysis of expected sarsa. In *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pp. 177–184. IEEE, 2009.

van Seijen, H., Fatemi, M., and Tavakoli, A. Using a logarithmic mapping to enable lower discount factors in reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Vieillard, N., Kozuno, T., Scherrer, B., Pietquin, O., Munos, R., and Geist, M. Leverage the average: an analysis of regularization in rl. *arXiv preprint arXiv:2003.14089*, 2020.

Wang, H., Zheng, S., Xiong, C., and Socher, R. On the generalization gap in reparameterizable reinforcement learning. In *International Conference on Machine Learning*, pp. 6648–6658, 2019.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Xu, Z., van Hasselt, H. P., and Silver, D. Meta-gradient reinforcement learning. In *Advances in neural information processing systems*, pp. 2396–2407, 2018.

Zhang, A., Ballas, N., and Pineau, J. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937*, 2018a.

Zhang, C., Vinyals, O., Munos, R., and Bengio, S. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018b.

Zhao, C., Siguad, O., Stulp, F., and Hospedales, T. M. Investigating generalisation in continuous deep reinforcement learning. *arXiv preprint arXiv:1902.07015*, 2019.

A. Appendix

A.1. Equivalence Proof for TD(0)

In this section we present the proof of Proposition 1.

Proof. Let ϕ be the sequence of parameters produced by Algorithm 1 when using discount factor γ_e, and added regularization function $\Psi(s,\theta) := \lambda\Big(\hat{V}_\theta(s)\Big)^2$, $\lambda := \frac{\gamma_e-\gamma}{2\gamma}$, reward scaling $\xi := \frac{\gamma_e}{\gamma}$, learning rate $\alpha_i' := \frac{\gamma}{\gamma_e}\alpha_i$ and same initial parameters θ_0. We will use induction to show $\theta_i = \phi_i, i = 1, 2, \ldots$.

The base case $\phi_0 = \theta_0$ follows immediately from the initialization. Assume $\phi_i = \theta_i$. We now prove for $i+1$.

We can rewrite i-th step of Algorithm 1 as

$$
\begin{aligned}
\theta_{i+1} &= \theta_i + \alpha_i \left(r + \gamma \hat{V}_{\theta_i}(s') - \hat{V}_{\theta_i}(s)\right) \nabla \hat{V}_{\theta_i}(s) \\
&\overset{(1)}{=} \phi_i + \alpha_i \left(r + \gamma \hat{V}_{\phi_i}(s') - \hat{V}_{\phi_i}(s)\right) \nabla \hat{V}_{\phi_i}(s) \\
&= \phi_i + \alpha_i \frac{\gamma}{\gamma_e}\big(\frac{\gamma_e}{\gamma} r + \gamma_e \hat{V}_{\phi_i}(s') - \frac{\gamma_e}{\gamma}\hat{V}_{\phi_i}(s)\big)\nabla \hat{V}_{\phi_i}(s) \\
&\overset{(2)}{=} \phi_i + \alpha_i'\big(\frac{\gamma_e}{\gamma} r + \gamma_e \hat{V}_{\phi_i}(s') - \hat{V}_{\phi_i}(s) + \hat{V}_{\phi_i}(s) - \frac{\gamma_e}{\gamma}\hat{V}_{\phi_i}(s)\big)\nabla \hat{V}_{\phi_i}(s) \\
&= \phi_i + \alpha_i' \left(\frac{\gamma_e}{\gamma} r + \gamma_e \hat{V}_{\phi_i}(s') - \hat{V}_{\phi_i}(s)\right) \nabla \hat{V}_{\phi_i}(s) - \alpha_i' \frac{\gamma_e - \gamma}{\gamma} \hat{V}_{\phi_i}(s) \nabla \hat{V}_{\phi_i}(s) \\
&= \phi_i + \alpha_i' \left(\frac{\gamma_e}{\gamma} r + \gamma_e \hat{V}_{\phi_i}(s') - \hat{V}_{\phi_i}(s)\right) \nabla \hat{V}_{\phi_i}(s) - \alpha_i' \nabla \left(\frac{\gamma_e - \gamma}{2\gamma}\Big(\hat{V}_{\phi_i}(s)\Big)^2\right) \\
&= \phi_{i+1}
\end{aligned}
$$

where equality (1) is due to the induction assumption and in (2) we defined $\alpha_i' := \alpha_i \frac{\gamma}{\gamma_e}$. □

A.2. Equivalence for SARSA and Expected SARSA

In this section we prove an equivalence for the Expected SARSA(0) algorithm (Algorithm 2), similarly to the proof for TD(0). Same arguments apply for the vanilla SARSA algorithm (where $\hat{V}_i(s')$ is replaced by $\hat{Q}_{\theta_i}(s', a')$).

Algorithm 2 Batch Expected SARSA(0)

Hyper-parameters: $\gamma \in [0, \gamma_e]$
Input: D, π
for $i = 0, 1, ..., N_{\text{iter}} - 1$ **do**
  Get uniformly random (s, a, r, s') from D
  $\hat{V}_i(s') := \sum_{a' \in \mathcal{A}} \pi(a'|s') \hat{Q}_{\theta_i}(s', a')$
  $\theta_{i+1} := \theta_i + \alpha_i [r + \gamma \hat{V}_i(s') - \hat{Q}_{\theta_i}(s, a)] \nabla \hat{Q}_{\theta_i}(s, a)$
end for

Algorithm 3 Batch Expected SARSA(0) with activation regularization

Hyper-parameters: $\lambda \in \mathbb{R}^+$ (regularization factor), $\xi \in \mathbb{R}^+$ (global reward scaling)
Input: D, π
for $i = 0, 1, ..., N_{\text{iter}} - 1$ **do**
  Get uniformly random (s, a, r, s') from D
  $\hat{V}_i(s') := \sum_{a' \in \mathcal{A}} \pi(a'|s') \hat{Q}_{\phi_i}(s', a')$
  $\phi_{i+1} := \phi_i + \alpha_i' [\xi r + \gamma_e \hat{V}_i(s') - \hat{Q}_{\phi_i}(s, a)] \nabla \hat{Q}_{\phi_i}(s, a) - \alpha_i' \nabla \left(\lambda \Big(\hat{Q}_{\phi_i}(s, a)\Big)^2\right)$.
end for

Proposition 2. *Let $\theta_1, \theta_2, ..$ be the parameters produced by Algorithm 2. If Algorithm 3 is run with initial parameters $\phi_0 := \theta_0$, step-size $\alpha_i' := \frac{\gamma}{\gamma_e}\alpha_i$, reward scaling $\xi := \frac{\gamma_e}{\gamma}$ and regularization factor $\lambda := \frac{\gamma_e - \gamma}{2\gamma}$ then it produces the same sequence of parameters, i.e, $\phi_k = \theta_k, k = 0, 1, ...$*

Proof. We prove by induction. The base case $\phi_0 = \theta_0$ follows immediately from the initialization.

Induction step: Assume $\phi_i = \theta_i$. We now prove for $i + 1$.

Using the induction assumption we can rewrite i-th step of Alg. 2 as

$$\begin{aligned}
\theta_{i+1} &= \theta_i + \alpha_i\Big(r + \gamma\hat{V}_{\theta_i}(s') - \hat{Q}_{\theta_i}(s,a)\Big)\nabla\hat{Q}_{\theta_i}(s,a) \\
&= \phi_i + \alpha_i\Big(r + \gamma\hat{V}_{\phi_i}(s') - \hat{Q}_{\phi_i}(s,a)\Big)\nabla\hat{Q}_{\phi_i}(s,a) \\
&= \phi_i + \alpha_i\frac{\gamma}{\gamma_e}\left(\frac{\gamma_e}{\gamma}r + \gamma_e\hat{V}_{\phi_i}(s') - \frac{\gamma_e}{\gamma}\hat{Q}_{\phi_i}(s,a)\right)\nabla\hat{Q}_{\phi_i}(s,a) \\
&\overset{(1)}{=} \phi_i + \alpha_i'\left(\frac{\gamma_e}{\gamma}r + \gamma_e\hat{V}_{\phi_i}(s') - \hat{Q}_{\phi_i}(s,a) + \hat{Q}_{\phi_i}(s,a) - \frac{\gamma_e}{\gamma}\hat{Q}_{\phi_i}(s,a)\right)\nabla\hat{Q}_{\phi_i}(s,a) \\
&= \phi_i + \alpha_i'\left(\frac{\gamma_e}{\gamma}r + \gamma_e\hat{V}_{\phi_i}(s') - \hat{Q}_{\phi_i}(s,a)\right)\nabla\hat{Q}_{\phi_i}(s,a) - \alpha_i'\frac{\gamma_e - \gamma}{\gamma}\hat{Q}_{\phi_i}(s,a)\nabla\hat{Q}_{\phi_i}(s,a) \\
&= \phi_i + \alpha_i'\left(\frac{\gamma_e}{\gamma}r + \gamma_e\hat{V}_{\phi_i}(s') - \hat{Q}_{\phi_i}(s,a)\right)\nabla\hat{Q}_{\phi_i}(s,a) - \alpha_i'\nabla\left(\frac{\gamma_e - \gamma}{2\gamma}\Big(\hat{Q}_{\phi_i}(s,a)\Big)^2\right) \\
&= \phi_{i+1},
\end{aligned}$$

where in (1) we defined $\alpha_i' := \alpha_i\frac{\gamma}{\gamma_e}$. □

A.3. The Equivalence m-step TD Prediction

In this section we will introduce a version of the equivalence for m-step TD updates.

The m-step TD update is defined as

$$\theta \leftarrow \theta + \alpha_i\left(\sum_{\tau=0}^{m-1}\gamma^\tau r_\tau + \gamma^m\hat{V}_\theta(s_n) - \hat{V}_\theta(s)\right)\nabla\hat{V}_\theta(s). \tag{2}$$

.

Proposition 3. *The semi-gradient m-step TD update step (2) is equivalent to the following update step*

$$\theta \leftarrow \theta + \alpha_i'\left(\sum_{\tau=0}^{m-1}\gamma^\tau\xi r_\tau + \gamma_e^m\hat{V}_\theta(s_m) - \hat{V}_\theta(s)\right)\nabla\hat{V}_\theta(s) - \alpha_i'\nabla\left(\lambda\Big(\hat{V}_\theta(s)\Big)^2\right), \tag{3}$$

where $\alpha_i' := \alpha_i\frac{\gamma^m}{\gamma_e^m}$ is a modified step size, $\xi = \frac{\gamma_e^m}{\gamma^m}$ is a global reward scaling, and $\lambda = \frac{\gamma_e^m - \gamma^m}{2\gamma^m}$ is a regularization factor.

Proof. We can rewrite the update of (2) as:

$$\begin{aligned}
&\theta + \alpha_i\left(\sum_{\tau=0}^{m-1}\gamma^\tau r_\tau + \gamma^m\hat{V}_\theta(s_m) - \hat{V}_\theta(s)\right)\nabla\hat{V}_\theta(s) \\
&= \theta + \alpha_i\frac{\gamma^m}{\gamma_e^m}\left(\sum_{\tau=0}^{m-1}\gamma^\tau\frac{\gamma_e^m}{\gamma^m}r_\tau + \gamma_e^m\hat{V}_\theta(s_m) - \frac{\gamma_e^m}{\gamma^m}\hat{V}_\theta(s)\right)\nabla\hat{V}_\theta(s) \\
&= \alpha_i\frac{\gamma^m}{\gamma_e^m}\left(\sum_{\tau=0}^{m-1}\gamma^\tau\frac{\gamma_e^m}{\gamma^m}r_\tau + \gamma_e^m\hat{V}_\theta(s_m) - \hat{V}_\theta(s) + \left(1 - \frac{\gamma_e^m}{\gamma^m}\right)\hat{V}_\theta(s)\right)\nabla\hat{V}_\theta(s)
\end{aligned}$$

Denoting $\alpha'_i := \alpha_i \frac{\gamma^m}{\gamma_e^m}$ we can write

$$= \alpha'_i \left(\sum_{\tau=0}^{m-1} \gamma^\tau \frac{\gamma_e^m}{\gamma^m} r_\tau + \gamma_e^m \hat{V}_\theta(s_m) - \hat{V}_\theta(s) \right) + \alpha'_i \left(1 - \frac{\gamma_e^m}{\gamma^m} \right) \hat{V}_\theta(s) \nabla \hat{V}_\theta(s)$$
$$= \alpha'_i \left(\sum_{\tau=0}^{m-1} \gamma^\tau \frac{\gamma_e^m}{\gamma^m} r_\tau + \gamma_e^m \hat{V}_\theta(s_m) - \hat{V}_\theta(s) \right) - \alpha'_i \nabla \left(\frac{\gamma_e^m - \gamma^m}{2\gamma^m} \left(\hat{V}_\theta(s) \right)^2 \right)$$

□

A.4. The Equivalence for the LSTD Algorithm

In this section we will introduce a version of the equivalence for the LSTD algorithm in the linear case.

Assume linear representation $\hat{V}_\theta(s) := \phi(s)^\top \theta$. The input is a set of transitions $\{(s_i, a_i, r_i, s'_i)\}_{i=1}^N$ We use the L_2 regularized LSTD algorithm with a guidance discount factor $\gamma \leq \gamma_e$ The algorithm output is $\theta := A^{-1}b$, where $A := \frac{1}{N} \sum_{i=1}^N \phi(s_i)(\phi(s_i) - \gamma\phi(s'_i))^\top + \lambda I$, and $b := \frac{1}{N} \sum_{i=1}^N r_i \phi(s_i)$.

We can re-write A as follows

$$A = \frac{1}{N} \sum_{i=1}^N \phi(s_i)(\phi(s_i) - \gamma\phi(s'_i))^\top + \lambda I$$
$$= \frac{1}{N} \sum_{i=1}^N \phi(s_i)(\phi(s_i) - \gamma_e\phi(s'_i))^\top + (\gamma_e - \gamma) \frac{1}{N} \sum_{i=1}^N \phi(s_i)\phi^\top(s'_i) + \lambda I$$

This shows that using a small discount $\gamma < \gamma_e$ in LSTD is equivalent to using a high discount γ_e and adding an *activation regularization*, $(\gamma_e - \gamma) \frac{1}{N} \sum_{i=1}^N \phi(s_i)\phi^\top(s'_i)$.

As we saw for TD(0), in the case of orthonormal features (as in the tabular case with uniform visitation), we have an exact equivalence to an L_2 regularization term.

The exact same derivation can be done for the LSTDQ algorithm (Lagoudakis & Parr, 2003).

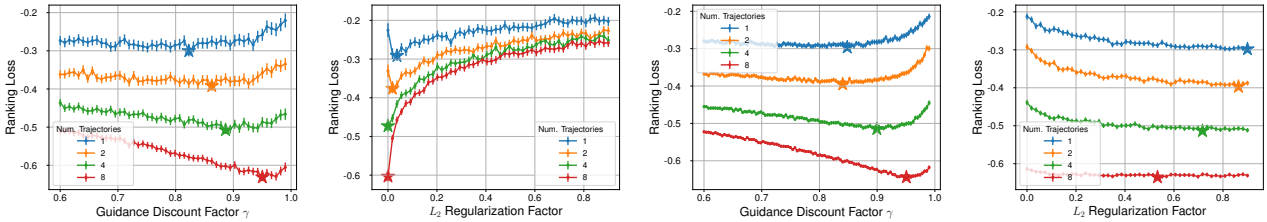
A.5. Tabular Experiments - Additional Details

A.5.1. GridWorld environment details.

We constructed a 4×4 GridWorld environment. For each instance of the MDP, a randomly chosen 'goal' state is assigned to a high reward mean of 1, while in all other states the reward mean is drawn uniformly from $[-0.5, 0.5]$. We assign the same reward mean for all actions at a given state. The instantaneous reward signal is drawn from Gaussian with standard deviation 0.1 and the state's reward mean. The available actions at each state are {'left', 'right', 'up', 'down', 'stay'}. If the move is not valid, then the agent remains in the same state. Otherwise, the agent moves to the new sate with probability p_s, or otherwise stays in the current state. The probabilities $p_s, \forall s \in \mathcal{S}$ are drawn uniformly from $[0, 1]$ when the MDP is created. In this problem there is no terminal state.

A.5.2. Evaluation method details.

We study the performance of the policy learned after on episode of approximate policy iteration. In the first stage of the episode, a batch of transitions is collected. Second, we run a batch policy evaluation algorithm to estimate the Q-function of the data collecting policy. Third, we derive the greedy policy w.r.t. Q, denoted π. The performance of π is measured by a loss function which is the L_1 distance of the value of the learned policy V^π to the value of the optimal policy V^*, $\|V^\pi - V^*\|_1$, where the values are computed with the true model and γ_e. We repeated the experiment for different numbers of samples collected in each episode. The results were averaged over 1000 repetitions.



(a) TD(0), Ranking loss vs. discount factor, several numbers of trajectories

(b) TD(0), Ranking loss vs. L_2 regularization factor, several numbers of trajectories

(c) LSTD, Ranking loss vs. discount factor, several numbers of trajectories

(d) LSTD, Ranking loss vs. L_2 regularization factor, several numbers of trajectories

Figure 6. **Tabular experiments with the ranking loss.** Loss vs. regularization factor for different regularizers, averaged over 1000 MDP instances. In each figure, the curves correspond to different number of samples per episode. The star shapes mark the minimum of the curve. Error bars represent 95% confidence interval.

A.5.3. TD(0) Expected-SARSA Algorithm Details.

In both algorithms we use large number of TD-iterations (5000) on the data set, where in each iterations we randomly sample a transition from the data set. We use a large number of iterations since we are interested in evaluating the final performance and not the convergence rate. The value function (or Q-function in Expected SARSA) is initialized with zero values. The learning rate is $\alpha_i := 500/(1000 + i)$, where i is the iteration index.

Procedure for Augmenting the Mixing Time of a Markov Process We describe the procedure we used for augmenting the transition probabilities matrix of a Markov Process to have a specific mixing time.

- Define the 'target' spectral gap according to the desired mixing time.
- Calculate the eigendecomposition of the transition matrix P.
- Force the desired spectral gap:
  - Re-scale the magnitude λ_2 to be according to the spectral gap.
  - For each other eigenvalue that now has a higher magnitude than λ_2, re-scale it to be $|\lambda_2|$.

A.6. Additional Tabular Experiments

A.6.1. Ranking Loss Evaluation

In Figure 6 we present the results with the ranking loss, corresponding to the results in Figure 3 of Section 4.1 with the L_2 loss. The ranking loss of the value estimation is defined by the negative Kendall's Tau correlation (Kendall, 1948) between the rankings of the estimated and true value functions (evaluated with the evaluation discount factor γ_e).

A.7. Complete Implementation Details of Mujoco Experiments

Our code uses the implementation of the TD3 and DDPG algorithms by Fujimoto et al. (2018). For completeness, we include here the full implementation details.

Critic Architecture

```
(state dim + action dim, 400)
ReLU
(action dim + 400, 300)
RelU
(300, 1)
```

Actor Architecture

Table 1. Hyper-parameters specification

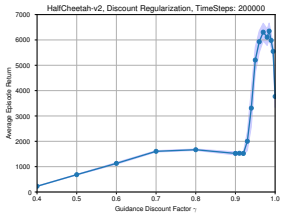
Hyper-parameter	Default Value	Grid
Critic Learning Rate	10^{-3}	-
Critic Regularization	None	$\lambda_{L_2} \cdot \|\|\theta\|\|^2, \lambda_{L_2} \in \{0., 0.005, 0.01, ...0.08\}$
Actor Learning Rate	10^{-3}	-
Actor Regularization	None	-
Optimizer	Adam	-
Target Update Rate (τ)	$5 \cdot 10^{-3}$	-
Batch Size	256	-
Iterations per time step	1	-
Discount Factor	0.999	$\gamma \in \{0.1, 0.2, ..., 0.9, 0.91, ..., 0.98, 0.985, 0.99, 0.995, 1.\}$
Reward Scaling	1.0	-
Normalized Observations	False	-
Gradient Clipping	False	-
Exploration Policy	$\mathcal{N}(0, 0.1)$	-

```
(state dim, 400)
ReLU
(400, 300)
RelU
(300, 1)
tanh
```

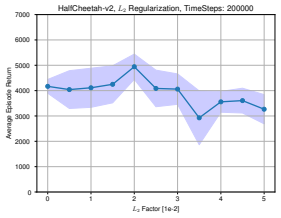
Each point is the parameter grid is averaged over 100 random seeds. The final policy is evaluated by averaging 10 episodes. The computing infrastructure for running the experiments used 4 GeForce GTX 1080 GPUs.

A.8. DDPG Algorithm Experiments

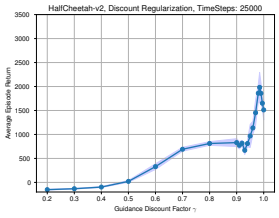
In Figure 7 we present results for the DDPG algorithm, corresponding to the results described in Figure 5 of Section 4.2 for the TD3 algorithm.



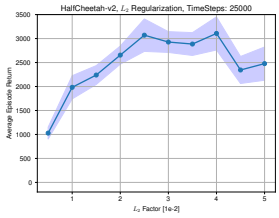
(a) HalfCheetah-v2, discount regularization, 2e5 steps



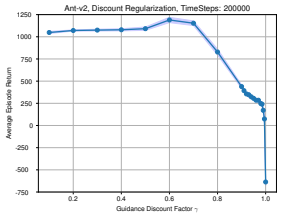
(b) HalfCheetah-v2, L_2 regularization, 2e5 steps



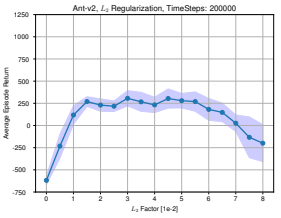
(c) HalfCheetah-v2, discount regularization, 25e3 steps



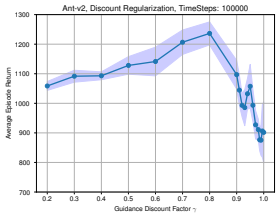
(d) HalfCheetah-v2, L_2 regularization, 25e3 steps



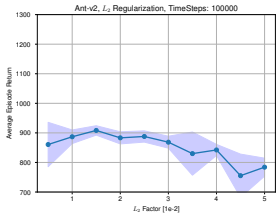
(e) Ant-v2, discount regularization, 2e5 steps



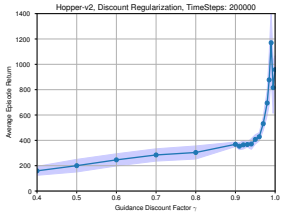
(f) Ant-v2, L_2 regularization, 2e5 steps



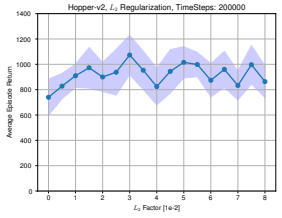
(g) Ant-v2, discount regularization, 1e5 steps



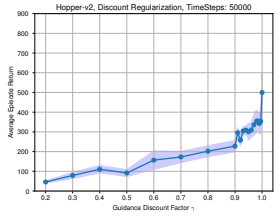
(h) Ant-v2, L_2 regularization, 1e5 steps



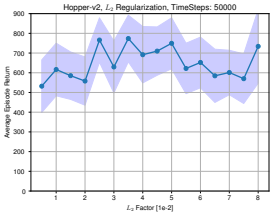
(i) Hopper-v2, discount regularization, 2e5 steps



(j) Hopper-v2, L_2 regularization, 2e5 steps



(k) Hopper-v2, discount regularization, 5e4 steps



(l) Hopper-v2, L_2 regularization, 5e4 steps

Figure 7. **Regularization in Mujoco experiments with limited data and DDPG algorithm.** Average total reward in evaluation episodes vs. regularization factor. Results are averaged over 20 simulations and 1000 evaluation episodes. Shaded area represent 95% confidence interval.