Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Adaptive Step-Size for Online Temporal Difference Learning

William Dabney and **Andrew G. Barto**
140 Governors Dr.
University of Massachusetts Amherst
Amherst, MA 01003

Abstract

The step-size, often denoted as α, is a key parameter for most incremental learning algorithms. Its importance is especially pronounced when performing online temporal difference (TD) learning with function approximation. Several methods have been developed to adapt the step-size online. These range from straightforward back-off strategies to adaptive algorithms based on gradient descent. We derive an adaptive upper bound on the step-size parameter to guarantee that online TD learning with linear function approximation will not diverge. We then empirically evaluate algorithms using this upper bound as a heuristic for adapting the step-size parameter online. We compare performance with related work including HL(λ) and Autostep. Our results show that this adaptive upper bound heuristic out-performs all existing methods without requiring any meta-parameters. This effectively eliminates the need to tune the learning rate of temporal difference learning with linear function approximation.

Introduction

One of the most pervasive parameters used throughout machine learning is the step-size parameter. The question of how far to step in the direction of the current update must be answered either explicitly in the form of a step-size parameter or implicitly by the learning algorithm itself. Because of the difficulty of the latter, the use of a fixed step-size parameter is common practice in many machine learning algorithms. Step-size is also highly influential on performance, so much so that it is a common practice to compare algorithms using performance with respect to a variety of step-sizes.

Where they exist, convergence guarantees for reinforcement learning and stochastic gradient descent algorithms ubiquitously rely on the step-size satisfying: $(\sum_{t=0}^{\infty} \alpha_t = \infty)$ & $(\sum_{t=0}^{\infty} \alpha_t^2 < \infty)$, where α_t denotes the step-size at time step t. Meeting these requirements will result in consistent learning, but can make convergence slower. On the other hand, choosing a fixed step-size that is too large can lead to very fast convergence in practice, but also has a greater chance of causing the process to diverge. The unfortunate irony is that often the best fixed step-size is the largest value that does not cause divergence. Thus, on one side of this threshold is greatly improved performance, and on the other side is a complete failure to learn. More discussion related to the problems with diverging function approximation can be found in Boyan and Moore (1995). Even when convergence does occur, performance is critically dependent on the choice of step-size.

The problems with divergence when using too large a step-size are especially severe in online temporal difference (TD) learning. For these algorithms, the step-size regulates the updating of values based on targets which are themselves estimates learned from previous updates. This *bootstrapping* process can exacerbate the effects of divergence by causing both target and the estimated current values to be increasingly poorly estimated.

Due to the critical importance of tuning step-size for a particular combination of learning algorithm and application domain, there has been extensive research on the topic of adapting the step-size. By far the most common approach to choosing a value for the step-size is to evaluate performance across a reasonable range of possible step-sizes and use the fixed value that leads to the lowest error or highest return. This method works well in practice, but is computationally expensive.

A class of approaches, which we refer to as *back-off strategies*, provides a fixed schedule of decreasing step-sizes. These are more theoretically sound and better at avoiding divergence than choosing a fixed step-size. One such schedule is given by the equation $\alpha_t = \frac{1}{t}$. This simple back-off strategy has been shown to be optimal for stationary data, where it matches the form for the unbiased estimation of the mean of the data (George and Powell 2006). For non-stationary data, a variety of other back-off strategies have been developed, such as the *search then converge* (STC) algorithm (Darken and Moody 1992). The underlying trait that all back-off strategies share is that they are fixed schedules. As such, they are unable to adapt the step-size if learning begins to diverge even after being tuned to the application domain.

Another approach, which we refer to as *adaptive strategies*, adapts the step-size based upon the prediction errors for the target (Sutton 1992b; Schraudolph 1999; George and Powell 2006; Hutter and Legg 2007; Mahmood et al. 2012). Some of these attempt to find an analytical solution for the optimal step-size, such as in the case of HL(λ) (Hutter and

Legg 2007). However, most approaches are themselves a form of dynamic programming or stochastic gradient descent, and as such must rely on a *meta-stepsize* parameter. The idea is that the algorithm will be much less sensitive to the meta-stepsize parameter than to the step-size itself. In practice these methods are still sensitive enough to the value of the meta-stepsize parameter that tuning must be done before they are used. The recently developed Autostep algorithm extends the Incremental Delta-Bar-Delta (IDBD) algorithm (Sutton 1992a), a gradient descent based adaptive step-size algorithm, to make it less sensitive to the value of its meta-parameters (Mahmood et al. 2012).

There are two drawbacks to existing adaptive strategies. First, they still require one or more meta-parameters that must be tuned ahead of time, although Autostep shows substantial progress on this point. Second, regardless of the meta-parameters, if the initial step-size is not chosen carefully they can still diverge. In this paper we derive upper and lower bounds for the optimal adaptive step-size for online temporal difference learning with linear function approximation. Then, because these bounds are dependent only on the experiences received by the learning algorithm, we are able to construct an adaptive step-size strategy based on the computation of approximate upper bounds on the optimal step-size.

The resulting algorithm is easily applicable to a wide variety of value function based learning algorithms, has per step computational complexity which is linear in the number of features , and is simple to understand and implement. We compare performance with a variety of existing backoff and adaptive strategies across four classic reinforcement learning problems. Despite its simplicity, our algorithm performs competitively or better than related approaches with tuned parameters, with the additional property that it prevents function approximation divergence without relying on any meta-parameters.

Derivation of Bounds

We now derive upper and lower bounds for the optimal adaptive step-size for an online temporal difference learning algorithm using linear function approximation. Note that we are assuming a scalar valued step-size, which does not offer as much potential for optimization as a vector containing a step-size for each input dimension, but is more commonly used in practice. With the exception of Autostep, all the algorithms compared in this paper use scalar valued step-sizes.

Let θ_t be the parameters for a linear function approximator over feature vector ϕ_t at time step t. Then, Equations 3 – 1 are the standard update equations for an online TD learning algorithm with eligibility traces, such as TD(λ) or Sarsa(λ). In our experiments we use Sarsa(λ) by assuming that the vectors ϕ, e, and θ are all of size $|\mathcal{A}||X|$, where $|\mathcal{A}|$ is the number of actions and $|X|$ is the number of input features. This is a fairly common practice, but we take note of it because upon going from a purely mathematical algorithm to an actual implementation, the details of what these vectors represent becomes important.

At each step, the learning algorithm receives a new feature vector ϕ_t and a reward r_t. The prediction error is computed using Equation 1,

$$\delta_t = \gamma\theta_t^T\phi_{t+1} + r_t - \theta_t^T\phi_t, \tag{1}$$

eligibility traces are updated with Equation 2,

$$e_t = \gamma\lambda e_{t-1} + \phi_t, \tag{2}$$

and finally the parameters of the function approximator are updated by Equation 3,

$$\theta_{t+1} = \theta_t + \alpha\delta_t e_t. \tag{3}$$

This last step of updating the parameters is where the step-size, α, is used to control how large of a change the update can make to the parameters.

The goal is for the step-size to be large enough so the process quickly converges, but small enough to avoid overshooting the learning target. If α is too large, we would intuitively expect that if we recomputed δ_t it might actually have increased, instead of getting closer to zero. In this situation, the sign before and after the update would have changed.

We can capture this intuition by considering the prediction error after the update from Equation 3. This is done by substituting the new parameter values θ_{t+1}, into Equation 1. This results in Equation 4,

$$\begin{aligned}\delta_t' &= \gamma\theta_{t+1}^T\phi_{t+1} + r_t - \theta_{t+1}^T\phi_t,\\ &= \gamma(\theta_t + \alpha\delta_t e_t)^T\phi_{t+1} + r_t - (\theta_t + \alpha\delta_t e_t)^T\phi_t,\\ &= \delta_t + \alpha\delta_t e_t^T(\gamma\phi_{t+1} - \phi_t).\end{aligned} \tag{4}$$

Using δ_t', we can now analyze the effect of the one step of learning. If the magnitude of the error decreased without changing signs, $|\delta_t'| < |\delta_t|$ and $sign(\delta_t') = sign(\delta_t)$, then we have, by this limited measurement, made some improvement. However, if δ_t' is actually further away from zero or has changed sign, then our estimation has gotten worse, suggesting our value of α is too large. Finally, if $\delta_t = \delta_t'$ then we have neither improved nor gotten worse.

Based upon these simple observations, Inequality 5 defines the boundaries of a successful update (one where the approximation has not gotten worse). With some manipulation this inequality comes down to three possible outcomes.

$$\begin{aligned} 0 \leq & \quad \tfrac{\delta_t'}{\delta_t} & \leq 1 \\ 0 \leq & \quad 1 + \alpha e_t^T(\gamma\phi_{t+1} - \phi_t) & \leq 1 \\ -1 \leq & \quad \alpha e_t^T(\gamma\phi_{t+1} - \phi_t) & \leq 0 \end{aligned} \tag{5}$$

Thus, when $e_t^T(\gamma\phi_{t+1} - \phi_t) = 0$, the result of this update will have no effect on δ_t,

$$e_t^T(\gamma\phi_{t+1} - \phi_t) = 0 \Leftrightarrow \delta_t' = \delta_t. \tag{6}$$

When $e_t^T(\gamma\phi_{t+1} - \phi_t) > 0$, the inequality forces $\alpha = 0$ in order to both satisfy the inequality and the requirement that $\alpha \in [0, 1]$. This is equivalent to ignoring the update completely,

$$e_t^T(\gamma\phi_{t+1} - \phi_t) > 0 \Leftrightarrow \frac{-1}{e_t^T(\gamma\phi_{t+1} - \phi_t)} \leq \alpha \leq 0$$
$$\Leftrightarrow \alpha = 0.$$

Finally, when $e_t^T(\gamma\phi_{t+1} - \phi_t) < 0$,

$$e_t^T(\gamma\phi_{t+1} - \phi_t) < 0 \Leftrightarrow \frac{-1}{e_t^T(\gamma\phi_{t+1} - \phi_t)} \geq \alpha \geq 0,$$
$$\Leftrightarrow 0 \leq \alpha \leq \frac{-1}{e_t^T(\gamma\phi_{t+1} - \phi_t)}, \quad (7)$$

we get a bound on the value of α that will ensure that the approximation does not get worse , at least locally.

However, this upper bound only takes into account the effect the update has locally (i.e., the effect on δ_t). It does not take into account the effects this update will have on other areas of the feature space. To get a tighter upper bound we can consider the effects that the update would have on all the previously computed deltas, $\delta_i \ \forall i \leq t$. This can be done as a straightforward extension to the derivation by considering the effect of the update on other prediction errors. This allows us to define a theoretically tighter upper bound on the optimal step-size at time step t, given in Equation 8,

$$\alpha_t^* \leq \min_{0 \leq i \leq t} \frac{-\delta_i}{\delta_t e_t^T(\gamma\phi_{i+1} - \phi_i)}. \quad (8)$$

Next, we look at deriving a lower bound on the optimal step-size. Notice that to compute the minimum over the whole set would require storing a vector for each step the agent has taken. Instead, in Equation 11 we use the Cauchy-Schwarz inequality to transform these inequalities to produce a new lower bound on α_t^*, the optimal step-size at time t. We start with the least upper bound and manipulate it into a maximum over inner products, Equation 10. Then, we are able to apply Cauchy-Schwarz, giving Inequality 11. This results in a lower bound given by Inequality 12, where L_t is given by Equations 13 & 14.

$$\alpha_t^* = \min_{0 \leq i \leq t} \frac{-\delta_i}{\delta_t e_t^T(\gamma\phi_{i+1} - \phi_i)}, \quad (9)$$
$$= \min_{0 \leq i \leq t} \left(\frac{\delta_t e_t^T(\gamma\phi_{i+1} - \phi_i)}{-\delta_i}\right)^{-1},$$
$$= \left(\max_{0 \leq i \leq t} \frac{\delta_t e_t^T(\gamma\phi_{i+1} - \phi_i)}{-\delta_i}\right)^{-1},$$
$$= \left(\max_{0 \leq i \leq t} | < \delta_t e_t, \frac{1}{\delta_i}(\gamma\phi_{i+1} - \phi_i) > |\right)^{-1},$$
$$\frac{1}{\alpha_t^*} = \max_{0 \leq i \leq t} | < \delta_t e_t, \frac{1}{\delta_i}(\gamma\phi_{i+1} - \phi_i) > |, \quad (10)$$
$$\leq \max_{0 \leq i \leq t} \|\delta_t e_t\| \cdot \|\frac{1}{\delta_i}(\gamma\phi_{i+1} - \phi_i)\|, \quad (11)$$
$$= \|\delta_t e_t\| \max_{0 \leq i \leq t} \|\frac{1}{\delta_i}(\gamma\phi_{i+1} - \phi_i)\|,$$
$$= \|\delta_t e_t\| L_t \Leftrightarrow,$$
$$\alpha_t^* \geq \frac{1}{\|\delta_t e_t\| L_t}. \quad (12)$$

$$L_t = \max[L_{t-1}, \|\frac{1}{\delta_t}(\gamma\phi_{t+1} - \phi_t)\|] \quad (13)$$
$$L_{-1} = 0 \quad (14)$$

These bounds are based on the assumption that we are concerned with divergence with respect to experiences received so far, as opposed to unknown future experiences. Based on this assumption, we can prove[1] that Inequality 8 gives a tight upper bound for the optimal adaptive step-size.

Application of Bounds

Based upon the above derivation, computing the true optimal step-size for step t would require storing the vector $\frac{(\gamma\phi_{i+1} - \phi_i)}{\delta_i}$ for all $i \leq t$, and would then require computing the constraints imposed by each previous time step and finding the resulting minimum value for the step-size. Instead, we used the heuristic of taking the minimum upper bound observed thus far as an approximation to the minimum over the whole set of constraints. This results in Equation 15, which requires $O(|\phi|)$ computational cost per time step and only the additional storage needed to do the computation:

$$\alpha_t = \min[\alpha_{t-1}, |e_t^T(\gamma\phi_{t+1} - \phi_t)|^{-1}] \quad (15)$$
$$\alpha_0 = 1.0$$

This update to the step-size should be done before the parameters are updated at each step where $e_t^T(\gamma\phi_{t+1} - \phi_t) < 0$. Finally, although we experiment with different settings of α_0 to illustrate performance, it is not necessary, either in theory or in practice, to use anything other than 1.0. Thus, Equation 15 does not depend on any meta-parameters.

Related Work - Back-off Strategies

George and Powell (2006) provide a thorough review of many back-off and adaptive strategies, referred to as deterministic and stochastic step-size rules respectively, but we review only those we use to obtain the experimental results. A more general form of the simple back-off strategy given in the introduction can be obtained by introducing a parameter τ. We call this the *simple back-off strategy*, which is given by $\alpha_t = \min[\alpha_0, \frac{\tau}{t}]$, where $\tau \in \mathbb{N}$ and t is the current time step. This strategy allows for a consistent decay that begins after a delay modulated by τ.

The Generalized Harmonic Stepsizes (*GHC*) algorithm, $\alpha_t = \alpha_0 \frac{\tau}{\tau + t - 1}$, provides the ability to directly set the rate of convergence using different values of τ (George and Powell 2006). The last back-off strategy that we will compare against is the Search then Converge (*STC*) algorithm (Darken and Moody 1992). This strategy does exactly what its name implies: early in the learning process it allows larger step-sizes and then past a certain point the step-sizes quickly converge. All three strategies have parameters that must be tuned, and are highly sensitive to their values.

[1] Proof provided in appendix.

Related Work - Adaptive Strategies

Out of the many adaptive strategies in the literature, we chose two based on their reduced reliance on meta-parameters. The first, HL(λ), is a completely parameter-free method (Hutter and Legg 2007). The second, Autostep, is the state-of-the-art gradient descent approach to adapting step-size with additional heuristics added that share the same intuition as our approach for deriving bounds on the optimal step-size (Mahmood et al. 2012).

HL(λ) Hutter and Legg (2007) derive an equation for adaptive step-sizes in online TD learning with discrete state spaces. The temporal difference update equation is comparable to the Sarsa(λ) algorithm already discussed, with Equation 16 giving the step-size for time step t where $\delta_{s_{t+1},s}$ is the Kronecker delta:

$$\begin{aligned} \alpha_t(s, s_{t+1}) &= \frac{1}{N^t_{s_{t+1}} - \gamma e^t_{s_{t+1}}} \frac{N^t_{s_{t+1}}}{N^t_s}, \text{where} \quad (16) \\ N^{t+1}_s &= \lambda N^t_s + \delta_{s_{t+1},s}. \end{aligned}$$

For our comparisons, we adapted this algorithm to continuous state spaces. We used the same approach used for eligibility traces in continuous spaces. Our continuous state version of the HL(λ) algorithm is given by Equation 17,

$$\begin{aligned} \alpha_t &= \frac{1}{N_t^T \phi_{t+1} - \gamma e_t^T \phi_{t+1}} \frac{N_t^T \phi_{t+1}}{N_t^T \phi_t}, \text{where} \quad (17) \\ N_{t+1} &= \lambda N_t + \phi_t. \end{aligned}$$

HL(λ) uses the discounted state visitation counter, N^t_s, for state s, and the eligibility traces at time step t to compute the new step-size. Our extension encodes this same information but with respect to the input feature vector instead of the discrete state.

Autostep Mahmood et al. (2012) recently presented Autostep, an approach for adapting the learning rate for gradient descent algorithms using linear function approximation. It is the closest related work, although it is designed for the general class of gradient descent algorithms with linear function approximation, and not specifically for use in temporal difference learning. Importantly, the paper clearly states that they hope to apply it to temporal difference learning in the future and suggest that this will require changes.

The basic intuition behind Autostep is very similar to the intuition driving this work, but the approach is substantially different. Autostep uses a gradient descent approach to automatically tune the alpha parameter. This requires three parameters: α_0 is the initial value of α, μ is the meta-learning rate, and τ is a scaling value which signals approximately how many samples over which to average. In their paper they, Mahmood et al. find that values of $\mu = 0.01$ and $\tau = 10000$ work best. The algorithm depends on α_0 being set sufficiently small so as not to cause divergence of the function approximation. This seems to defeat one of the reasons to adapt the learning rate.

Experimental Results

To evaluate the effectiveness of using our derived upper bounds as a heuristic for adapting the step-size, we used the Sarsa(λ) TD learning algorithm with different methods for setting the step-size parameter α. All experiments used a Fourier basis of order 3 and are averages over 10 runs with standard deviation shown. We evaluated on the following classic reinforcement learning domains: Mountain Car, Acrobot, Puddle World, and Cart Pole. To simplify the process of experimenting on multiple domains with a variety of algorithms, we used the RL-Glue framework with its implementations for the domains (Tanner and White 2009), and the Sarsa(λ) with Fourier basis algorithm of Konidaris, Osentoski, and Thomas (2011).

Briefly, we describe the problem domains; for more in-depth descriptions see Boyan and Moore (1995), Sutton (1996), and Sutton and Barto (1998). Unless otherwise specified there is a -1 reward per step with a reward of 0 at the goal. The Mountain Car domain is the task of controlling an underpowered car which is stuck in a valley so that it reaches the top of the hill in front of it. It is fully observable, has two continuous state variables (the car's position and velocity) and three discrete actions (reverse, neutral, and forward). The Acrobot domain is the task of swinging up a two jointed rigid body; there are four continuous state variables corresponding to joint angles and velocities, and three discrete actions for applying positive, negative or neutral torque to the second joint. The Puddle World domain is a navigation task within a square room with large randomly generated puddles in it. There are two continuous state variables for the location in the room and four discrete actions (up, down, left and right). There is a negative reward that gets larger as the agent gets deeper into the puddle, a time step reward of -1 and reward of 0 at the goal. Finally, the Cart Pole domain is the task of keeping a stick, connected to a cart by a hinge, from falling over. There are four continuous state variables (cart location, cart velocity, angle of pole, and angular velocity of pole) and two discrete actions (move left or right). There is a reward of 1.0 for every step, and the episode ends when the pole reaches a ± 12 degree angle or the cart's position goes past ± 2.4.

We will refer to a particular combination of step-size strategy, meta-parameter setting (if applicable), and domain as a *trial*. We evaluated performance for each trial as follows. In each trial, the learning algorithm was trained online in the domain for 375 episodes. Every 25 episodes learning was frozen and the performance was evaluated for 10 episodes and the average total reward for those evaluation episodes was recorded. These evaluations were averaged together to get the average total reward for that trial. Each trial was run 10 times and the averages and standard deviations were used for our figures. Parameters used for all experiments were: $\gamma = 1.0$, $\lambda = 0.9$, and $\epsilon = 0.01$. One exception is that $\gamma = 0.99$ was used for experiments with the HL(λ) algorithm because a value of 1.0 causes the denominator in Equation 16 to vanish. This is a property of the original as well as our extended version of the algorithm.

For all these experiments we set α_0 in Equation 15 to the same value that the other methods used. However, unlike

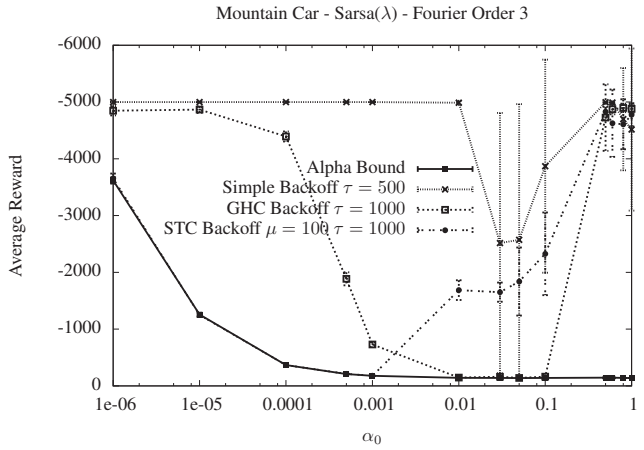

Figure 1: Comparing various back-off strategies with alpha bounds and vanilla (fixed step-size). Domain: Mountain Car

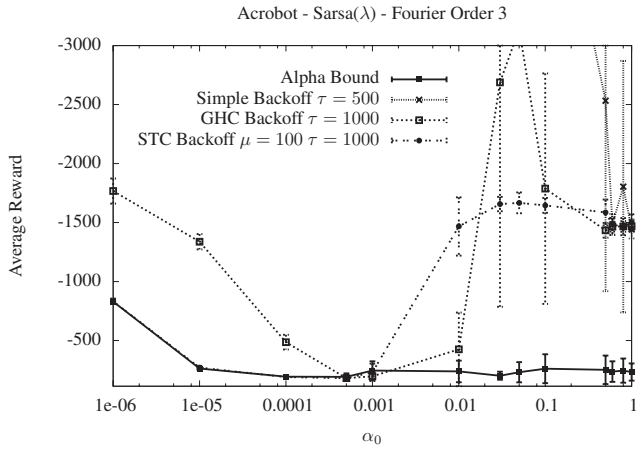

Figure 2: Comparing various back-off strategies with alpha bounds and vanilla (fixed step-size). Domain: Acrobot

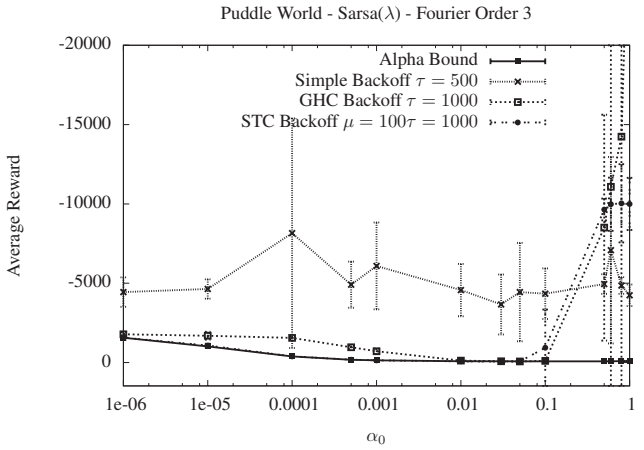

Figure 3: Comparing various back-off strategies with alpha bounds and vanilla (fixed step-size). Domain: Puddle World

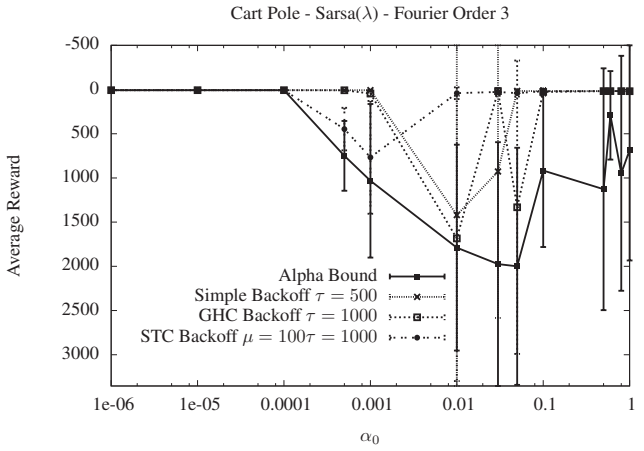

Figure 4: Comparing various back-off strategies with alpha bounds and vanilla (fixed step-size). Domain: Cart Pole

the related approaches, our method never resulted in function approximation divergence in any trial. Moreover, setting $\alpha_0 = 1.0$ resulted in performance that is comparable to that obtained with the best fixed step-size. Only in Cart Pole does our approach perform worse for larger initial step-sizes.

First, we consider the performance of the three back-off strategies we previously discussed. Figures 1–4 show how these perform compared with using the alpha bound to set the step-size. For the simple back-off strategy we experimented with values for τ (1, 100, 500, and 1000). For the GHC strategy we considered values for τ (1, 100, 500, 1000). Finally, for STC we experimented with values for τ (100, 1000, 10000) and for μ (1, 100, 1000). The results shown are for the meta-parameter values that performed best. In general we did not find that the back-off strategies were much more effective than a fixed step-size. In some of the domains, such as Mountain Car in Figure 1, they performed worse than a fixed step-size. However, in some domains, such as Puddle World, they showed improved performance over the fixed step-size approach. However, universally they were outperformed by the alpha bound.

Next, we examine performance using the alpha bounds compared with the adaptive strategies, using a fixed step-size to show a baseline for performance. For the adaptive strategies we show the performance of Autostep and HL(λ) in Figures 5–8. Notice that Autostep is not able to prevent function approximation divergence when the initial step-size is set too high. Despite this drawback, Autostep performs very well when the step size is small enough. With small initial step-sizes, it is able to adapt the step-size and gradually increase it, resulting in better performance than a fixed step-size of the same initial value. One of the strong points of the algorithm is its robustness with respect to the choice of meta-parameter. We show results with $\mu = 0.01$ and $\tau = 10000$, which performed best among the values we tested.

Our results show that the HL(λ) algorithm loses some per-

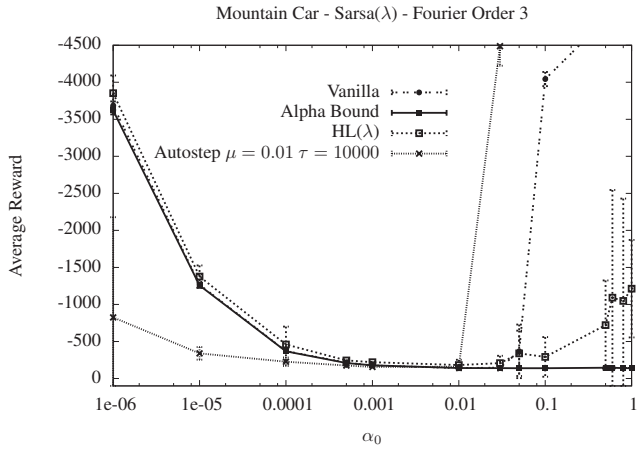

Figure 5: Comparing adaptive strategies with alpha bounds and vanilla (fixed step-size). Domain: Mountain Car

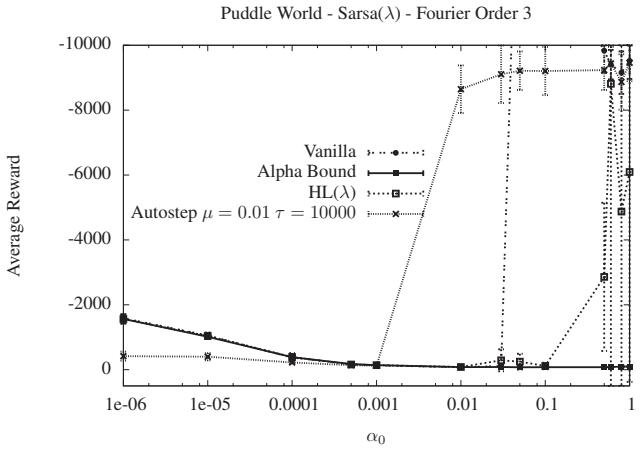

Figure 7: Comparing adaptive strategies with alpha bounds and vanilla (fixed step-size). Domain: Puddle World

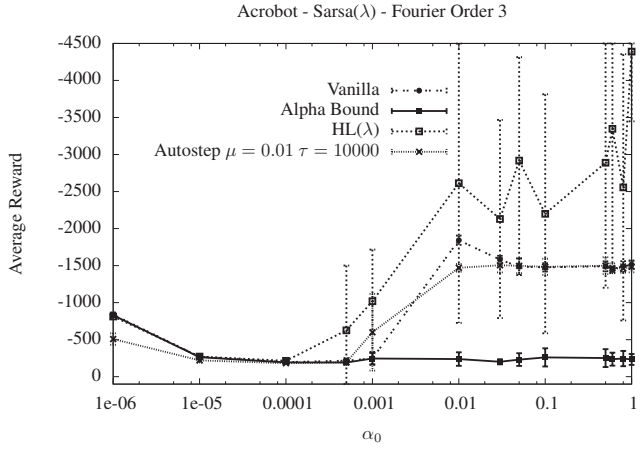

Figure 6: Comparing adaptive strategies with alpha bounds and vanilla (fixed step-size). Domain: Acrobot

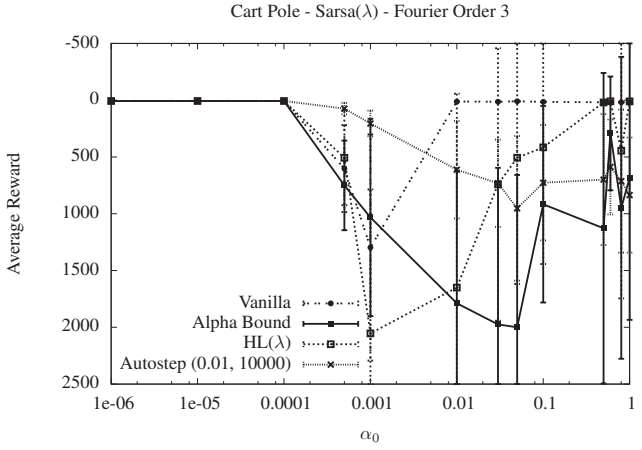

Figure 8: Comparing adaptive strategies with alpha bounds and vanilla (fixed step-size). Domain: Cart Pole

formance in continuous domains, where as Hutter and Legg found that it outperformed TD(λ) in all situations in their discrete domain. In our experiments, HL(λ) did as well or better than vanilla Sarsa(λ) in all but the Acrobot domain, where it frequently diverged. Divergence was still a problem in the other domains but it happened less frequently with HL(λ) than with a fixed step-size as the initial step-size got closer to 1.0. However, in these comparisons we found that using the upper bounds previously derived to adapt the step-size continued to perform best.

Conclusion

We have derived new upper and lower bounds for the optimal step-size for online TD learning with linear function approximation. We showed that a simple algorithm using these bounds to adapt the step-size online outperforms existing methods, and unlike all existing approaches, completely prevents function approximation divergence. For future research, extending our derivation of upper and lower bounds on scalar step-sizes to the vector of step-sizes case is the most promising direction. This would allow step-sizes to be adapted individually for each input dimension, potentially leading to better algorithms for adaptive step-sizes. Another promising direction for future research is to develop algorithms that make use of the lower bound to do something akin to the search then converge algorithm, where the step-size is initially computed based upon the upper bounds and gradually converges to the lower bound.

Appendix: Proof of Upper Bound Tightness

Formally we define the step-size *upper bound* as some positive value, greater than or equal to the true optimal step-size, such that any larger step-size will result in a change of sign in prediction error or an increase in the error magni-

tude for some previously experienced transition and reward. We explicitly assume that the only information available is the sequence $\{\phi_t, a_t, r_t, e_t\}_{t\geq 0}$ tuples received at each time step, the learning algorithm, and parameters used $\{\alpha_t, \gamma, \lambda\}$. Thus, we are assuming that no model of the underlying MDP is provided. If we have access to the MDP, that model could be exploited to improve the step-size upper bounds, but in this situation we would be unlikely to be applying model-free reinforcement learning in the first place.

Theorem 1. $\alpha_t^* \leq \min_{0\leq i\leq t} \frac{-\delta_i}{\delta_t e_t^T(\gamma\phi_{i+1}-\phi_i)}$ *is a tight upper bound for non-zero step-sizes.*

Proof. Assume to the contrary that there exists $0 < \alpha < \alpha_t^*$ such that α is an upper bound for step-size. Then, using α_t^* for the value function update at time t will cause prediction error for some observed transition, $0 \leq i \leq t$, to increase in absolute magnitude or change signs and α will not cause such a change. Then: $\alpha < \alpha_t^* \Rightarrow \exists i : 0 \leq i \leq t$ such that $\frac{\delta_i'}{\delta_i} > 1$ or $\frac{\delta_i'}{\delta_i} < 0$ when α_t^* is used for the update, and $0 \leq \frac{\delta_i'}{\delta_i} \leq 1$ when α is used for the update.

First we expand on exactly what we mean by δ_i and δ_i'. They are the prediction error for the ith experienced step using the parameters at time t (before the update using α_t^*) and $t+1$ (after the update using α_t^*) respectively. Formally:

$$\begin{aligned}
\delta_i &= \gamma\theta_t^T\phi_{i+1} + r_i - \theta_t^T\phi_i, \\
\delta_i' &= \gamma\theta_{t+1}^T\phi_{i+1} + r_i - \theta_{t+1}^T\phi_i, \\
&= \gamma(\theta_t + \alpha_t^*\delta_t e_t)^T\phi_{i+1} + r_i - (\theta_t + \alpha_t^*\delta_t e_t)^T\phi_i, \\
&= \delta_i + \alpha_t^*\delta_t e_t^T(\gamma\phi_{i+1} - \phi_i). \qquad (18)
\end{aligned}$$

Now consider the condition we would expect if our assumption to the contrary holds, specifically the value of δ_i'/δ_i. Here, we have used j to denote the argmin of Equation 8. Notice that the second term in Equation 19 is exactly the ratio between two elements from the minimization.

$$\begin{aligned}
\frac{\delta_i'}{\delta_i} &= 1 + \frac{\delta_t}{\delta_i}\alpha_t^* e_t^T(\gamma\phi_{i+1} - \phi_i) \\
&= 1 - \frac{\delta_j e_t^T(\gamma\phi_{i+1} - \phi_i)}{\delta_i e_t^T(\gamma\phi_{j+1} - \phi_j)} \qquad (19)
\end{aligned}$$

Now, we have two possible cases that would allow our contrary assumed α to exist. First, if $\frac{\delta_i'}{\delta_i} < 0$ then the second term in Equation 19 must be greater than 1. This implies that there exists $\beta = \frac{-\delta_i}{\delta_t e_t^T(\gamma\phi_{i+1}-\phi_i)}$ such that $\beta < \alpha_t^*$. This is a contradiction because β is a member of the set of which α_t^* is taken to be the minimum.

Second, if $\frac{\delta_i'}{\delta_i} > 1$ then the second term in Equation 19 must be less than zero. This implies that the same β we were considering before is now less than zero, because α_t^* is taken to be greater than zero. Now, consider this possibility in the context of using our contrary assumed α as an update:

$$\begin{aligned}
0 \leq & \quad \tfrac{\delta_i'}{\delta_i} & \leq 1 \\
-1 \leq & \quad \alpha\tfrac{\delta_t}{\delta_i} e_t^T(\gamma\phi_{i+1} - \phi_i) & \leq 0 \\
\frac{-\delta_i}{\delta_t e_t^T(\gamma\phi_{i+1} - \phi_i)} \leq & \quad \alpha & \leq 0
\end{aligned}$$

Therefore, $\alpha = 0$, which again is a contradiction because we assumed that $\alpha > 0$. Thus, both possible cases lead to contradictions. Therefore, we conclude that no such lower upper bound, $0 < \alpha < \alpha_t^*$, exists. □

References

Boyan, J. A., and Moore, A. W. 1995. Generalization in reinforcement learning: Safely approximating the value function. *Advances in Neural Information Processing Systems 7*.

Darken, C., and Moody, J. 1992. Towards faster stochastic gradient search. *Advances in Neural Information Processing Systems 4*.

George, A. P., and Powell, W. B. 2006. Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine Learning* 65:167–198.

Hutter, M., and Legg, S. 2007. Temporal difference updating without a learning rate. *Advances in Neural Information Processing Systems*.

Konidaris, G. D.; Osentoski, S.; and Thomas, P. 2011. Value function approximation in reinforcement learning using the fourier basis. *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*.

Mahmood, A. R.; Sutton, R. S.; Degris, T.; and Pilarski, P. M. 2012. Tuning-free step-size adaptation. *International Conference on Acoustics, Speech, and Signal Processing*.

Schraudolph, N. N. 1999. Local gain adaptation in stochastic gradient. *Proceedings of the International Conference on Artificial Neural Networks* 569–574.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

Sutton, R. S. 1992a. Adapting bias by gradient descent: An incremental version of delta-bar-delta. *Proceedings of the National Conference on Artificial Intelligence* 171–176.

Sutton, R. S. 1992b. Gain adaptation beats least squares. *Proceedings of the Seventh Yale Workshop on Adaptive Learning Systems* 161–166.

Sutton, R. S. 1996. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Neural Information Processing Systems 8* 1038–1044.

Tanner, B., and White, A. 2009. RL-Glue : Language-independent software for reinforcement-learning experiments. *Journal of Machine Learning Research* 10:2133–2136.