Generalizing Value Estimation over Timescale

Craig Sherstan [1] **James MacGlashan** [2] **Patrick M. Pilarski** [1]

Abstract

General value functions (GVFs) are an approach to representing models of an agent's world as a collection of predictive questions. A GVF is defined by: a policy, a prediction target, and a timescale. Traditionally predictions for a given timescale must be specified by the engineer and each timescale learned independently. Here we present γ-nets, a method for generalizing value function estimation over timescale, allowing a given GVF to be trained and queried for any fixed timescale. The key to our approach is to use timescale as one of the network inputs. The prediction target for any fixed timescale is then available at every timestep and we are free to train on any number of timescales. We present preliminary results on a simple test signal.

1. Value Functions and Timescale

Reinforcement learning (RL) studies algorithms in which an agent learns to maximize the amount of reward it receives over its lifetime. A key method in RL is the estimation of *value* — the expected cumulative sum of discounted future rewards (called the *return*). In loose terms this tells an agent how good it is to be in a particular state. The agent can then learn a *policy* — a way of behaving — which maximizes the amount of reward received.

Sutton et al. (2011) broadened the use of value estimation by introducing general value functions (GVFs), in which value estimates are made of other sensorimotor signals, not just reward. GVFs can be thought of as representing an agent's model of itself and its environment as a collection of questions about future sensorimotor returns; a predictive representation of state. A GVF is defined by three elements: 1) the policy, 2) the *cumulant* (the sensorimotor signal to be predicted), and 3) the prediction timescale, γ. Considering a simple mobile robot, examples of GVF questions include "How much current will my motors consume over the next 3 seconds if I spin clockwise?" or "How long until my bump sensor goes high if I drive forward?"

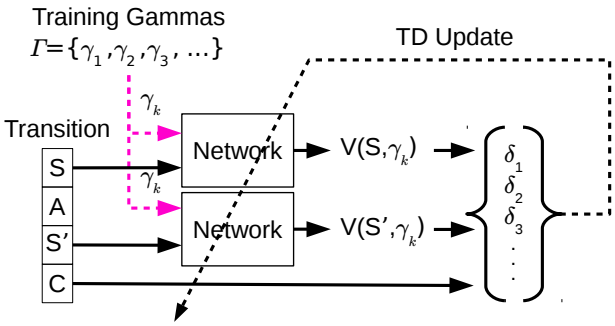

Figure 1. **γ-nets**. Values are estimated by providing state and timescale, γ, as inputs to the network parameterized by weights $\mathbf{w}$. An agent in state S takes action A and transitions to state S' receiving the new target signal C. The agent selects a set of timescales Γ on which to train and for each $\gamma \in \Gamma$ computes values $V(S,\gamma;\mathbf{w})$ and $V(S',\gamma;\mathbf{w})$ (note that there is only a single network, but it must be called twice to compute each V). For each γ, the TD error is calculated according to $\delta = C+\gamma V(S',\gamma;\mathbf{w})-V(S,\gamma;\mathbf{w})$. The TD errors are then collected and used to update $\mathbf{w}$ using a chosen TD learning algorithm, such as TD(λ) or GTD.

This paper focuses on generalizing value estimation over timescale. Modeling the world at many timescales is seen as a key problem in artificial intelligence (Sutton, 1995; Precup et al., 1998). Further, there is evidence that humans and other animals make estimates of reward and other signals at numerous timescales (Tanaka et al., 2016). Our work here can be seen as directly connected to the concept of *nexting*, in which animals and people make large numbers of predictions of sensory input at many, short-term, timescales (Gilbert, 2006). Modayil et al. (2014) demonstrated the concept of nexting using GVFs on a mobile robot. However, until now, value estimation has, in general, been limited to a single fixed timescale. That is, for each desired timescale, a discrete and unique predictor was learned. However, there are many situations where we may desire to have value estimates of the same cumulant over many different timescales. For example, consider an agent driving a car. Such an agent may make numerous predictions about the likelihood of col-

[1]Department of Computing Science, University of Alberta, Edmonton, Canada [2]Cogitai, Inc., United States. Correspondence to: Craig Sherstan <sherstan@ualberta.ca>, Patrick M. Pilarski <pilarski@ualberta.ca>.

Accepted at the FAIM workshop "Prediction and Generative Modeling in Reinforcement Learning", Stockholm, Sweden, 2018.

liding with various objects in its vicinity. The agent needs to consider both the risk of collisions in the near term and far term and the relevance of each may change with the speed of the car. If the engineer knew which timescales would be needed ahead of time they could design them into the system, but this is not the case for complex settings.

Here we present a novel class of algorithms which enables the explicit learning and inference of value estimates for any valid fixed discount. The key insights to our approach are: 1) the timescale can be treated as an input parameter for inference and learning and 2) the estimated bootstrapped prediction target for any fixed timescale is available at every timestep. We provide a first simple demonstration of our approach in predicting the return of a standard square pulse over many different timescales.

1.1. Related Work

Schaul et al. (2015) generalized value estimation across goals, and their corresponding policies, by providing a goal embedding vector as input to the value estimator. Xu et al. (2018) created a deep reinforcement learning algorithm which automatically adapts the meta-parameters, including the discount, of a reinforcement learning algorithm by gradient descent. The goal of their algorithm was to automatically determine the best return to use for learning as the agent learned. They found that providing the meta-parameters, including γ, as input to both their value and policy networks significantly improved performance in learning policies on 57 Atari games. Similarly, our algorithm provides the timescale as input to the function approximation network.

2. Background

We model the agent's interaction with its environment as a Markov Decision Process. At each timestep t the agent, in state $S_t \in \mathcal{S}$, takes action $A_t \in \mathcal{A}$ according to policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ and transitions to state $S_{t+1} \in \mathcal{S}$ according to the transition probability $p(\cdot|S_t, A_t)$. In the traditional RL setting the agent receives a reward $R_{t+1} \equiv R(S_t, A_t, S_{t+1}) \in \mathbb{R}$. The agent's goal is to learn a policy which maximizes the amount of cumulative reward it receives in the future, which is defined as the return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots. \tag{1}$$

The term $\gamma \in [0, 1]$ is referred to by several names in the literature including the timescale, the continuation function and the discount; it represents the amount of emphasis applied to future rewards and is the focus of this paper. In the case of GVFs we simply substitute our signal of interest, the cumulant, C for reward, R.

A value estimate is simply the expectation of the return:

$$V_\pi(s) = \mathbb{E}_\pi\Big[G_t | S_t = s\Big]. \tag{2}$$

Temporal difference (TD) learning is a common class of algorithms used in RL for learning an approximation of value (Sutton & Barto, 1998). Estimation weights are typically trained by stochastic gradient descent using the TD error:

$$\delta_t = C_{t+1} + \gamma V(S_{t+1}) - V(S_t).$$

While simple domains can be represented using tabular lookup, complex settings in which the state space is very large or infinite must use function approximation (FA) methods to estimate the value as $V(s; \mathbf{w})$, where $\mathbf{w}$ is a set of weights parameterizing the network. Function approximation has the advantage that states are not treated independently, but rather, a learning step updates related states as well allowing for generalization across state-space.

3. Generalizing over γ

Our goal is to be able to predict the value function for any discount factor γ. To achieve that goal, we propose γ-nets: a neural network architecture for value functions that operates not only on the state, but also the desired target discount factor γ_k (see Figure 1). On each transition the network is trained on many arbitrary $\gamma_k \in \Gamma$ values. Thus, the γ-net learns to generalize over arbitrary γ_k values.

Generating the error function for a γ-net is also straightforward. For any single $\gamma_k \in \Gamma$, the usual TD error is modified to:

$$\delta_{t;\gamma_k} = C_{t+1} + \gamma_k V(S_{t+1}, \gamma_k) - V(S_{t+1}, \gamma_k). \tag{3}$$

The total gradient can then be summed over all $\gamma_k \in \Gamma$ and applied to update the network.

Choosing Γ must be done with care. A naive approach might uniformly sample $\gamma_k \in [0, 1)$[1]. However, value functions change non-linearly with γ. To illustrate this property, consider that γ can be viewed as the probability of continuation, allowing us to derive the expected number of timesteps until termination of the return as (see Sherstan (2015) for a derivation):

$$\tau = \frac{1}{1 - \gamma}. \tag{4}$$

This relationship is non-linear for large values of γ (Figure 4). Thus, naively drawing γ_k from a uniform distribution would tend to favor very short timescales. Conversely,

[1]If $\gamma = 1$ value estimates are infinite unless the return is guaranteed to terminate as in the episodic case.

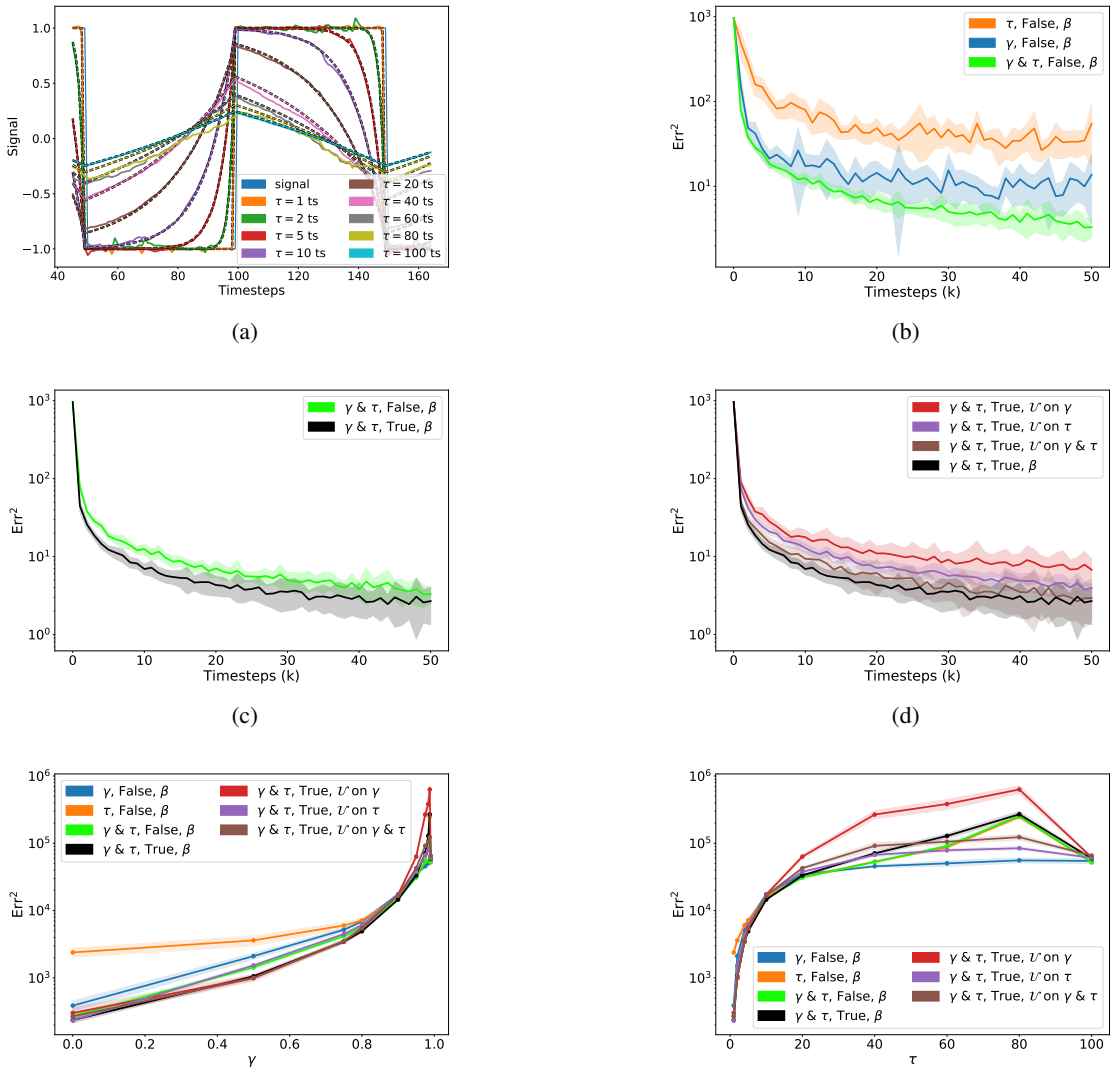

Figure 2. Experimental results. **a**) Predictions (solid) against the true return (dashed) after 50k ts using both γ and τ as input, prescaling the cumulant and using the β-distribution described in Section 4. For display purposes all predictions are normalized by $(1-\gamma)$. We see good accuracy for both of the bookend timescales $\tau = \{1, 100\}$. Shorter timescale estimates show decent accuracy, but error increases with longer timescales. This is to be expected from the distribution used for training (Figure 5) which has very little probability of selecting timescales in the region of 60 or 80 ts for training. **b–f**) The error of various configurations is compared. The legend tuple indicates (inputs, prescaling, distribution used for timescale selection). For each experiment, runs of 50k ts were averaged over 40 seeds. Network weights were saved every 1k ts. The networks were evaluated at $\tau = \{1, 2, 4, 5, 10, 20, 40, 60, 80, 100\}$ and 95% confidence intervals are indicated by shading. **b–d**) The squared error was computed against the ideal return and summed across all prediction timescales. The squared error of each timescale is given comparable weighting by normalizing by $(1-\gamma)^2$. **b**) Comparing effects of input representation. **c**) Comparing the use of cumulant pre-scaling. **d**) Comparing distributions used for gamma selection. **e, f**) The unnormalized squared errors for each evaluated timescale summed over time with respect to γ (**e**) and τ (**f**).

drawing uniformly from τ would put little emphasis on short timescales. While the best method for selecting γ_k for training is outside the scope of this paper, we provide some comparisons in our experiments.

The representation of timescale used for input to the network may affect the network's ability to represent different timescales. The γ scale compresses long timescales but spreads short ones and in the τ scale we have the opposite effect. Thus, providing both γ and τ as input may allow for good discrimination at all timescales.

Finally, the magnitude of near-term and far-term returns is very different. To prevent far-term returns from dominating

the function approximation we may need to normalize the returns in some way (van Hasselt et al., 2016). One approach to normalization in our setting is to prescale each cumulant by $(1-\gamma)$. Predictions can then be reconstructed by dividing the network output by this term.

4. Experiments

Our prediction target was a continuous square wave 100 timesteps in length with a magnitude of $\{-1, 1\}$ (Figure 2a). Normalized inputs were tilecoded (Sutton & Barto, 1998) with 20 tilings of width 1.0, 20 tilings of width 0.5 and 30 tilings of width 0.1. Tiling positions were randomly shifted by small amounts at the time of initialization for each run. Value estimates were computed using linear function approximation on the output of the tilecoding and the final layer of weights were updated using TD(0) (Sutton & Barto, 1998) (The algorithm used for this experiment is given in Algorithm 1). For training, at each timestep six timescales were used; we always trained on $\tau = \{1, 100\}$ and drew four timescales from a chosen distribution. We compared using uniform distributions on both γ and τ and mixing half from each, but found that a beta distribution gave better results (Figure 2d). The beta distribution sampled values of τ with support $\in [1, 100]$ and parameters $\alpha = 1, \beta = 4$ (Figure 5). For network input we used the current timestep $\in [0, 99]$ and the timescale expressed as either γ, τ or both. We also evaluated the impact of prescaling the cumulants as discussed in Section 3.

Results are shown in Figure 2. Here the network is evaluated for numerous timescale values. The squared error for each is normalized by $(1-\gamma)^2$ to give equal weighting to all timescales and the errors are summed across all timescales. The behavior of the prediction accuracy is different when γ or τ are used as network input and the lowest error is found when both are provided (Figure 2b). Prescaling the cumulant further lowered the aggregated error across the selected timescales (Figure 2c). This combination also gives excellent performance on very low timescales such as $\gamma = 0$. This is good as we expect that, in general, shorter timescales may be more important than long ones as state representation. Additionally, we compared between the beta distribution and uniform distributions on γ and τ and a mixed distribution in which two timescale values were drawn from each of the uniform distributions (Figure 2d). The beta distribution gave the lowest error, but the mixed uniform distribution was nearly as good. Our results, although not conclusive or statistically significant in many cases, suggest that when using a linear function approximation network, better results are achieved by prescaling the cumulants and using both γ and τ as inputs. Further consideration must be given to how the training timescales are selected.

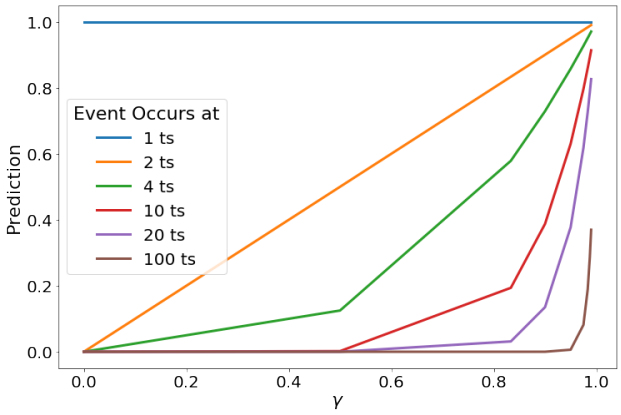

Figure 3. For each series a single event, value=1.0, occurs at some time in the future. Different timescales show different responses to these events. This suggests that different prediction timescales can enhance an agent's state representation.

5. Using γ-nets

GVFs have typically been studied as a form of predictive representation of state. A prediction at a given timescale compresses the expected return into a single number, which results in a loss of information about the form of the return. That is, an infinite number of returns can result in the same expected return from a given state. By looking at many different timescales of the value additional information about the return can be recovered, e.g. is the return many small near-term values or one large far-term value? Figure 3 illustrates the different returns of events at various distances in the future. Being able to make predictions about the environment at various timescales thus provides an agent with a richer representation of its world and its effect on its world. This ought to enable an agent to learn better policies.

However, it is not clear how our system could be integrated into a larger agent. Consider that it may be useful to the agent at time t to have predictions of a given signal over many timescales. First, there has to be some mechanism by which the agent could determine what values of γ would be most useful to it. Secondly, it has to determine how many γ values to use. If this number changes from timestep to timestep then the agent has to be able to handle making decisions with varying length inputs. Thirdly, for each desired timescale a forward pass must be made on the network. All of these issues are likely solvable, but they do not fit well within the current architectures in popular use. Further, it is not clear that having predictions at arbitrary timescales would be better than simply using a fixed set of timescales.

6. Future Work

The method we have described in this paper has thus far been limited to the fixed discounting case. However, one of the

key generalizations of GVFs is to support state-dependent (Sutton et al., 2011) or even transition-dependent discounting functions $\gamma_{t+1} \equiv \gamma(S_t, A_t, S_{t+1})$ (White, 2017). This allows GVFs to be far more expressive in terms of what they estimate. The generalized return is then given as

$$\begin{aligned} G_t &= C_{t+1} + \gamma_{t+1} C_{t+2} + \gamma_{t+1}\gamma_{t+2} C_{t+3} + \dots \\ &= \sum_{k}^{\infty} (\prod_{j=1}^{k} \gamma_{t+j}) C_{t+k+1}. \end{aligned}$$

Extending our method to support transition-dependent discounting is clearly an important next-step in this work.

The successor representation (SR) (Dayan, 1993) predicts counts of future state visitation. It is a method of factorizing value estimation into two components, one which captures the dynamics of the environment under a particular policy and discount function (the SR), and a one-step reward prediction. It has been suggested by several authors that the SR may have numerous uses including that of transfer learning (Barreto et al., 2017; Ma et al., 2018), option discovery (Machado et al., 2018) and accelerating GVF learning in continual learning settings (Sherstan et al., 2018). However, the SR is dependent on a policy and a discount function. Ma et al. (2018) have already generalized the SR across goals (and their induced policies); a natural next-step is to generalize the SR over timescale.

7. Conclusion

We have presented γ-nets, a simple technique for generalizing value estimation across timescale. This technique allows a system to make predictions for values of any timescale within the training regime of the network. This ability may be useful in several ways such as in predictive representations of state — modeling the world as a collection of predictions about future sensorimotor signals. In complex environments, such as the real world, complete models are not feasible, thus, being able to query for any timescale provides a means of making the model more compact and expressive. While we have demonstrated a mechanism for generalizing value estimates over fixed discounting functions, we have not demonstrated its usefulness. In particular, additional work is needed to show: 1) that predictions of varying discounts are useful in policy learning, and 2) that our system of generalization can be effectively combined with policy learning.

8. Acknowledgments

Conception and initial work for this paper was performed by Sherstan while an intern at Cogitai.

References

Barreto, A, Dabney, W, Munos, R, Hunt, J, Schaul, T, Silver, D, and van Hasselt, H. Transfer in Reinforcement Learning with Successor Features and Generalised Policy Improvement. In *Lifelong Learning: A Reinforcement Learning Approach Workshop @ICML*, Sydney, Australia, 2017.

Dayan, P. Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation*, 5(4):613–624, 1993.

Gilbert, D. *Stumbling on Happiness*. Knopf Press, 2006.

Ma, C, Wen, J, and Bengio, Y. Universal Successor Representations for Transfer Reinforcement Learning. *arXiv*, 1804.03758:1–4, 2018.

Machado, M C., Rosenbaum, C, Guo, X, Liu, M, Tesauro, G, and Campbell, M. Eigenoption Discovery Through The Deep Successor Representation. In *International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

Modayil, J, White, A, and Sutton, R S. Multi-Timescale Nexting in a Reinforcement Learning Robot. *Adaptive Behavior*, 22(2):146–160, 2014.

Precup, D, Sutton, R S, and Singh, S. Multi-time Models for Temporally Abstract Planning. *Advances in Neural Information Processing Systems (NIPS)*, 10(1995):1050–1056, 1998.

Schaul, T, Horgan, D, Gregor, K, and Silver, D. Universal Value Function Approximators. *The International Conference on Machine Learning (ICML)*, pp. 1312–1320, 2015.

Sherstan, C. *Towards Prosthetic Arms as Wearable Intelligent Robots*. PhD thesis, University of Alberta, 2015.

Sherstan, C, Machado, M C., and Pilarski, P M. Accelerating Learning in Constructive Predictive Frameworks with the Successor Representation. *arXiv*, 1803.09001, 2018.

Sutton, R S. TD Models: Modeling the World at a Mixture of Time Scales. In *The International Conference on Machine Learning (ICML)*, volume 53, pp. 531–539, 1995.

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

Sutton, R S, Modayil, J, Delp, M, Degris, T, Pilarski, P M, White, A, and Precup, D. Horde: A Scalable Real-time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction. In *The International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 2, pp. 761–768, Taipei, Taiwan, 2011.

Tanaka, S C., Doya, K, Okada, G, Ueda, K, Okamoto, Y, and Yamawaki, S. Prediction of Immediate and Future Rewards Differentially Recruits Cortico-Basal Ganglia Loops. *Behavioral Economics of Preferences, Choices, and Happiness*, 7(8):593–616, 2016.

van Hasselt, H, Guez, A, Hessel, M, Mnih, V, and Silver, D. Learning Values Across Many Orders of Magnitude. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 4287–4295, Barcelona, Spain, 2016.

White, M. Unifying Task Specification in Reinforcement Learning. *arXiv*, 1609.01995, 2017.

Xu, Zhongwen, van Hasselt, Hado, and Silver, David. Meta-Gradient Reinforcement Learning. *arXiv*, 1805.09801, 2018.

A. Additional Figures

Table 1. Expected Timesteps

γ	τ
0	1
0.5	2
0.8	5
0.9	10
0.95	20
0.975	40
0.983	60
0.9875	80
0.99	100

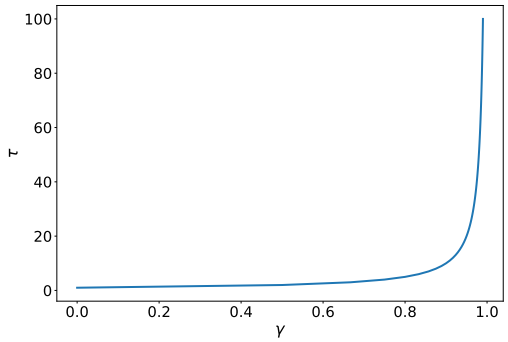

Figure 4. Non-linear relationship between discount γ and prediction length in expected timesteps.

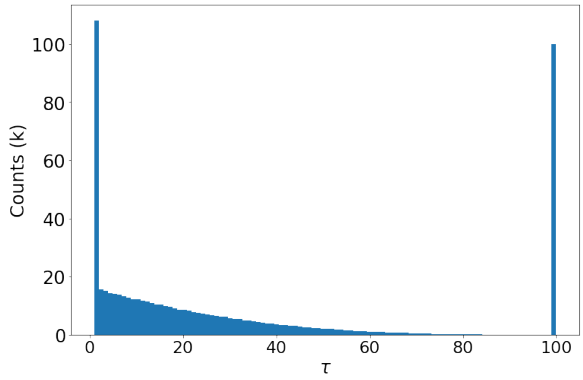

Figure 5. Histogram of timescales used for training in Section 4 over 100000 timesteps. Six timescales were used on each timestep; $\tau = \{1, 100\}$ were trained on each timestep and an additional four were selected from a beta distribution over τ with support on $[1, 100]$ and parameters $\alpha = 1, \beta = 4$.

B. Increased Representational Power

One of the advantages of TD algorithms is that they allow the agent to bootstrap estimation of the return from its existing estimates. However, this limits a single predictor to capturing only returns with geometric discounting as in Equation (1). By combining predictions at several timescales it is possible to capture returns of different shapes such as the example shown in Figure 6.

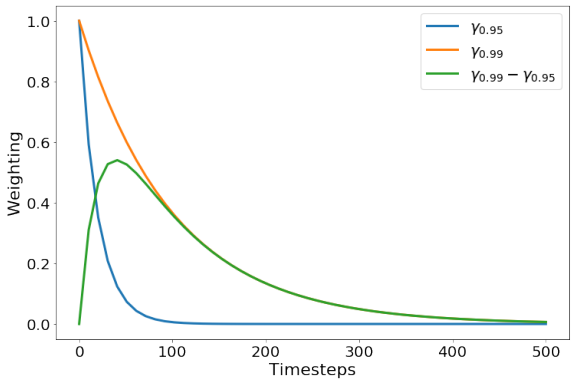

Figure 6. Capturing non-geometric returns by taking the difference of two predictions at different timescales.

C. Training Algorithm

Algorithm 1 Generalization over γ with TD(0)

Input: Feature representation $\phi \in \mathbb{R}^n$, policy π, and step-size α
Output: Vector $\mathbf{w}$.

Initialize $\mathbf{w} \in \mathbb{R}^n$ arbitrarily
while S' is not terminal **do**
  Observe state S, take action A selected according to $\pi(S)$, and observe a next state S' and cumulant C
  Pick a set of γ_k to train on:
  $\Gamma \leftarrow \gamma SelectionFunction(Terminal = False)$
  $\boldsymbol{\Delta} \leftarrow \mathbf{0}$; Zeros vector, length n
  for γ_k in Γ **do**
    $\delta = C + \gamma_k \phi(S_{t+1}, \gamma_k)^\top \mathbf{w} - \phi(S_t, \gamma_k)^\top \mathbf{w}$
    $\boldsymbol{\Delta} \mathrel{+}= \delta \phi(S_t, \gamma_k)$
  end for
  $\mathbf{w} = \mathbf{w} + \alpha \boldsymbol{\Delta}$
end while
$\Gamma \leftarrow \gamma SelectionFunction(Terminal = True)$
for γ_k in Γ **do**
  $\delta = C - \phi(S_t, \gamma_k)^\top \mathbf{w}$
  $\boldsymbol{\Delta} \mathrel{+}= \delta \phi(S_t, \gamma_k)$
end for
$\mathbf{w} = \mathbf{w} + \alpha \boldsymbol{\Delta}$