Stable predictive representations with general value functions for continual learning

Matthew Schlegel, Adam White and Martha White
Department of Computer Science,
Indiana University at Bloomington
{mkschleg,adamw,martha}@indiana.edu

Abstract

The objective of continual learning is to build agents that continually learn about their world, building on prior learning. In this paper, we explore an approach to continual learning based on making and updating many predictions formalized as general value functions (GVFs). The idea behind GVFs is simple: if we can cast the task of representing predictive knowledge as a prediction of future reward, then computationally efficient policy evaluation methods from reinforcement learning can be used to learn a large collection of predictions while the agent interacts with the world. We explore this idea further by analyzing how GVF predictions can be used as predictive features, and introduce two algorithmic techniques to ensure the stability of continual prediction learning. We illustrate these ideas with a small experiment in the cycle world domain.

1. Introduction

Continual learning is an approach to achieving the long-standing goals of artificial intelligence research: building agents that can know a lot about the world and use that knowledge to flexibly achieve goals. The main idea behind continual learning is to leverage an unending stream of experience—a life-time—continually building on prior learning. In this paper, we explore an approach to continual learning based on making and updating many predictions about the data generated by an agent's interaction with the world.

This predictive approach has a long tradition in machine learning, starting with predictive state representations (PSRs). A PSR summarizes everything the agent knows about the world as a large collection of predictions about the probability to different sequences of actions and observations occurring (Littman et al., 2001). Given a minimal set of core tests and their probability of success, the agent can generate the probability of success of any other test—the agent's state representation is entirely composed of predictions. A PSR uses a finite set of prediction to model partially observable tasks, rather than storing potentially infinite history of observed data. PSRs have several important limitations. Most notably, PSRs are restricted to predicting non-continuous observations (Singh et al., 2003; James and Singh, 2004; Wiewiora, 2005; McCracken and Bowling, 2005; Wolfe et al., 2005; Bowling et al., 2006; Wolfe and Singh, 2006), and algorithms for learning the predictions suffer from numerical issues limiting their application to small finite state domains. Extensions to continuous observations have been restricted to linear PSRs (Rosencrantz et al., 2004; Rudary et al., 2005; Wingate and Singh, 2006; Boots et al., 2011), using approaches from subspace identification which have issues with scalability.

Temporal difference networks (TD-nets) (Sutton et al., 2005) take the predictive approach a step further with several important innovations compared with PSRs. TD-nets enable learning from and about continuous observations. TD-nets also enable learning compositional predictions: making predictions about the outcome of other predictions. Like PSRs, TD-nets enable predictions to be used as state, while TD-nets with options allow predictions to be made about the outcomes of multi-step options (Rafols, 2006). There is still plenty of room to build upon the predictive approach to continual learning. TD-nets use a specialized learning rule that can diverge under off-policy sampling, and in practice the error in the predictions can oscillate wildly. In order to improve stability, Silver (2012) introduced a recurrent neural network variant of TD-nets using gradient TD to update the weights.

Recently, generalized value functions have been proposed as an alternative representation scheme to TD-nets. In particular, we can replace the scalar discount factor used to define conventional value functions with a state-based continuation function, and simply allow the reward function to be any observable signal. This simple generalization allows complex predictions to be learned using simple and robust value function learning algorithms from reinforcement learning, as opposed to the specialized learning rules required for TD-nets. Value function learning algorithms require only linear computation and storage per prediction, are

guaranteed to converge, are compatible with linear and non-linear function approximation, and are well suited to learning in non-stationary environments. A collection of GVFs can emulate predictive state representations by simply using each GVF's prediction as input to the feature construction process (White, 2015).

In this short paper, we introduce an explicit stabilization layer to a collection of GVFs, resulting in a simple and flexible architecture. The stabilization layer adapts parameters to produce stable features from inputted predictions, rather than adjusting the predictions themselves to improve stability. Decoupling these two roles ensures that predictions do not need to balance between accuracy and utility as a feature. We introduce an extension of the GTD(λ) algorithm to include clipped elastic regularization. We illustrate the utility of our architecture with a small experiment in the cycle world domain.

2. Problem formulation

We model the interaction between an agent and its environment as a Markov decision process $(\mathcal{S}, \mathcal{A}, \mathrm{T}, r)$, where $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ denotes the set of actions, and $\mathrm{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ encodes the one-step state transition dynamics. On each discrete time step $t = 1, 2, 3, ...$, the agent selects an action according to its behavior policy, $A_t \sim \mu(\cdot|S_t)$, and the environment responds by transitioning into a new state S_{t+1}.The agent cannot directly access the state, but instead observes a vector, $o_{t+1} \in \mathbb{R}^m$, that provides incomplete information about the current state.

On each transition the agent observes a *cumulant*, $Z_{t+1} \stackrel{\text{def}}{=} z(S_t, A_t, S_{t+1})$. The cumulant can be any real-valued signal that the agent can observe, such as a sensor reading or feedback generated by a person. The cumulant is used to define a multi-step prediction *target* $G_t \in \mathbb{R}$, where $G_t \stackrel{\text{def}}{=} \sum_{k=0}^{\infty} \left(\Pi_{j=1}^{k} \gamma_{t+j}\right) Z_{t+k+1}$. The continuation signal $\gamma_{t+1} \stackrel{\text{def}}{=} \gamma(S_t, A_t, S_{t+1})$ specifies the horizon of the prediction. Given this definition of G_t, we can specify a *general value function* (GVF) $v(s) = \mathbb{E}_\mu[G_t|S_t = s]$, where $v : \mathcal{S} \to \mathbb{R}$. We can cast the task of learning multi-step predictions as one of estimating a value function, where the prediction on time-step t is simply $V_t \approx v(S_t)$. Any prediction that can be represented with a TD-net can be expressed as one or more GVFs. Due to space restrictions we direct the reader to prior work detailing the specification and expressiveness of GVFs (Sutton et al., 2011; White, 2015).

Assuming a linear approximation of the value function, we can estimate these GVFs while interacting with the world using temporal difference learning algorithms such as the GTD(λ) algorithm (Sutton et al., 2009b,a; Maei et al., 2009; Maei, 2011):

$$
\begin{aligned}
\delta_t &= Z_{t+1} + \gamma_{t+1}\mathbf{w}_t^\top \mathbf{x}_{t+1} - \mathbf{w}_t^\top \mathbf{x}_t \\
\mathbf{e}_t &= \rho_t(\lambda_t \gamma_t \mathbf{e}_{t-1} + x_t) \qquad \triangleright\ \rho_t \stackrel{\text{def}}{=} \pi(S_t, A_t)/\mu(S_t, A_t) \text{ is the importance-sampling weight} \\
\mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha\delta_t\mathbf{e}_t - \alpha\gamma_{t+1}(1 - \lambda_{t+1})(\mathbf{e}_t^\top \mathbf{h}_t)\mathbf{x}_{t+1} \\
\mathbf{h}_{t+1} &= \mathbf{h}_t + \alpha_h[\delta_t\mathbf{e}_t - (\mathbf{x}_t^\top \mathbf{h}_t)x_t] \qquad \triangleright \text{ auxiliary weights,}
\end{aligned}
$$

where the feature vector $\mathbf{x}_t \in \mathbb{R}^d$ is constructed based on everything observable to the agent, and the prediction is linear in the features: $V_t \stackrel{\text{def}}{=} \mathbf{x}_t^\top \mathbf{w}$. The cumulant here plays the same role as the reward in conventional RL, but the cumulant is not necessarily maximized by the control policy μ. The objective of the agent is to learn many GVFs in parallel. Each GVF specifies a predictive question (indexed by $0 \leq i \leq n$): if the agent selected actions according to *target policy* $\pi^{(i)}$, what would be the expected future sum of cumulant values, $Z^{(i)}$, with a prediction horizon specified by $\gamma^{(i)}$.

3. Stabilizing predictive feature learning

We explore how a collection of GVFs can be connected to incorporate both predictive features and an explicit stabilization layer. The architecture consists of three repeated components: (1) predictions from the previous time step feed in as inputs to the construction of the state, (2) the stabilization layer nonlinearly transforms those predictions to produce features and add regularization; and (3) the predictions for the current time step are outputted (to then feed in as inputs for the next time step). The first part of the proposed stabilization layer performs an expansion with a nonlinear encoding that has outputs between zero and one, such as tile coding, Fourier basis or radial basis functions. This component provides a simple approach to manage the magnitude of the predictions: even if prediction begin to diverge, the encoding maps this value to bins with

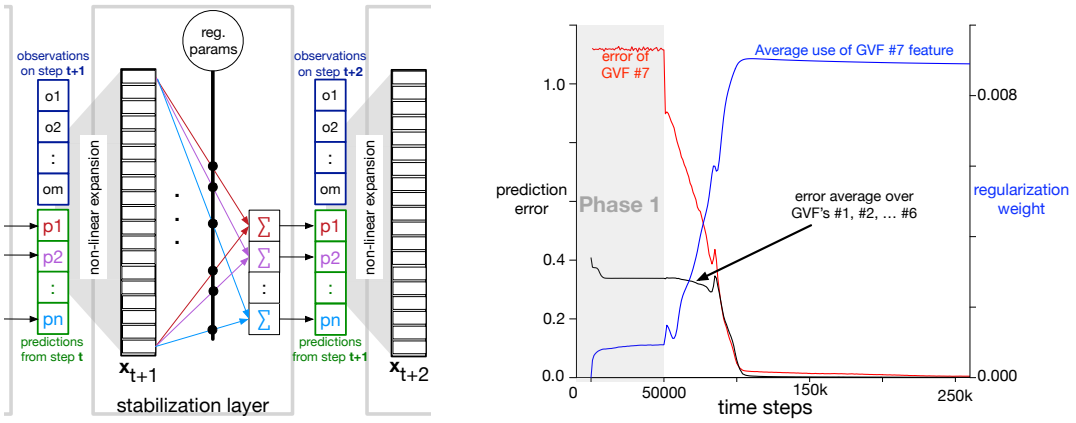

Figure 1 (LHS): The inputs pass through a nonlinear expansion, such as tile coding, producing the feature vector $\mathbf{x}$. Note the different time-step on the observations, and predictions used to construct $\mathbf{x}$. The feature vector is weighted linearly to produce the next set of predictions. An adaptive regularization weight, based on long-term statistical properties of both the predictions and observations, is added to the computation of each prediction.

Figure 1 (RHS): Cycle world results. Above we plot (1) the error averaged over of the one-step, two-step, up to six-step cycle world predictions in black, (2) the error of the prediction of *expected number of steps until observing a "1"* in red (GVF #7), and (3) the average of all the weights on the predictive features generated from GVF #7's prediction. During phase one of the experiment GVF #7 is not allowed to learn; it was set equal to a random number. Our regularization scheme prevented GVF #7's erratic prediction from causing the other predictions to destabilize. Nevertheless, the first six GVFs did not obtain good accuracy. In the second phase of the experiment, starting at step 50,000, we allowed GVF #7 to update. The regularization scheme then allowed the other GVFs to incorporate (through the construction of $\mathbf{x}$) the information summarized in GVF #7's prediction, and all GVFs converged to zero prediction error.

fixed magnitude. This encoding has the additional benefit of improving modeling capabilities, providing a nonlinear transformation that is more powerful than typical smooth activation functions.

The adaptive regularization component is our second line of defense against unstable learning, moderating highly variable predictions and removing less useful features. An adaptive regularization parameter can be set for each prediction, given a measure of learning variability or inaccuracy for each feature. We add a regularizer $R(\boldsymbol{H}\mathbf{w})$ to the objective, for regularization function $R : \mathbb{R}^d \to \mathbb{R}^+$ and $\boldsymbol{H} \in \mathbb{R}^{d\times d}$ a diagonal matrix with regularization weights $\eta_i \geq 0$ on the diagonal.

It is straightforward to modify the GTD(λ) updates to incorporate this regularizer. GTD(λ) minimizes the mean-squared projected Bellman error (MSPBE); to incorporate regularization, we simply need to add a regularizer to this objective for the updates. To add regularization, $\mathbf{w}$ is updated with the standard GTD(λ) update and then updated according to the given regularizer. For a smooth regularizer, the update is $\mathbf{w} \leftarrow \mathbf{w} - \alpha\boldsymbol{H}\nabla R(\boldsymbol{H}\mathbf{w})$ and for a non-smooth regularizer (Mahadevan et al., 2014), the update is $\mathbf{w} \leftarrow \text{prox}_{\alpha R(\boldsymbol{H}\cdot)}(\mathbf{w})$ where the proximal operator for function $\alpha R(\boldsymbol{H}\cdot)$ is $\text{prox}_{\alpha R(\boldsymbol{H}\cdot)}(\mathbf{w}) \stackrel{\text{def}}{=} \arg\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{u} - \mathbf{w}\|_2^2 + \alpha R(\boldsymbol{H}\mathbf{u})$.

We propose to use a clipped elastic net regularizer, which promotes removing problematic features. The clipped regularizer is defined as $c(\mathbf{w}) = \sum_{i=1}^d \eta_i \min(\nu|\mathbf{w}_i| + (1-\nu)\mathbf{w}_i^2, \epsilon)$, for some $\epsilon > 0$ and $\nu \in [0, 1]$. The clipped regularizer avoids the typical shrinkage properties of ℓ_p norms—which can incur significant bias—because values of $|\mathbf{w}_i| + \mathbf{w}_i^2$ above ϵ are not penalized additionally. Instead, particularly with the inclusion of the ℓ_1, the regularization mainly performs feature selection, zeroing less useful features. The ℓ_2 regularizer is strongly convex and can significantly speed the convergence rate. The proximal operator[1] is

$$\text{prox}_{\alpha c}(\mathbf{w})_i = \begin{cases} \mathbf{w}_i & : \nu|\mathbf{w}_i| + (1-\nu)\mathbf{w}_i^2 \geq \epsilon \\ \text{sign}(\tilde{\mathbf{w}}_i)\max(|\tilde{\mathbf{w}}_i| - \nu\alpha\eta_i, 0) & : \text{otherwise, with } \tilde{\mathbf{w}}_i = (1 - 2(1-\nu)\alpha\eta_i)\mathbf{w}_i \end{cases}$$

We can similarly regularize the second set of weights $\mathbf{h}$, which is learned using the same (potentially poor) features. This regularization, however, changes the MSPBE. By adding an ℓ_2 regularizer to the update for $\mathbf{h}$, the effect is to modify the covariance matrix $E[\mathbf{x}_t\mathbf{x}_t^\top]$ in the MSPBE to $E[\mathbf{x}_t\mathbf{x}_t^\top] + \eta\mathbf{I}$. The solution to this modified MSPBE is still equal to the TD fixed point, but the learned $\mathbf{h}$ is more stable.

1. Theoretical development for proximal operators is typically for convex regularizers; the clipped regularizer, however, is nonconvex. Nonetheless, the proximal gradient update can be applied because our proximal operator has a unique solution (Yu et al., 2015).

4. Experiments in cycle world

We conducted our empirical study on the cycle world, a domain known to exhibit unstable learning for one-step TD-nets. There are six states arranged in a ring, and on each time-step the agent deterministically transitions clock-wise to the next state. The observation of "1" is available in one of the states, in all other states, the observation is "0". The agent makes six predictions: the probability of observing "1" on the next time step, in two time steps, and so on. These predictions can in principle be represented by a conventional one-step TD-net without history[2], but in practice learning results in instability and inaccurate predictions (Tanner and Sutton, 2005).

We specified six GVFs, one for each of the cycle world predictions. The one-step prediction used a constant continuation $\gamma^{(1)}_{t+1} = 0$, and $Z^{(1)}_{t+1} = o_{t+1}$. The two-step prediction requires a compositional GVF: $\gamma^{(2)}_{t+1} = 0$, and $Z^{(2)}_{t+1} = V^{(1)}_{t+1}$, where $V^{(1)}_{t+1}$ is the prediction made by the one-step GVF. The three-step prediction was also compositional, depending on the two step prediction: $\gamma^{(3)}_{t+1} = 0$, and $Z^{(3)}_{t+1} = V^{(2)}_{t+1}$. The other the three GVF's are specified similarly. In addition we added a multi-step (non-zero γ) prediction, which is easy to learn and useful for obtaining good prediction accuracy on the other six GVF questions (through the feature vector, see Section 3). The final prediction encodes the expected number of steps until the "1" will be observed: $\gamma^{(7)}_{t+1} = 0$ if o_{t+1} equals "1" and $\gamma^{(7)}_{t+1} = 1$ otherwise, and a constant cumulant $Z^{(7)}_{t+1} = 1$. The feature vector was constructed by independently tile coding each prediction, resulting in a binary $\mathbf{x}$ with 514 components (one bit for the observation and a bias unit), of which 258 were always active. The cycle world is a uncontrolled (single-action) Markov reward process so the predictions are learned on-policy.

We performed a simple experiment to illustrate the simplicity of posing GVF questions, and to highlight the stability properties of our architecture. The experiment was divided into two phases. During the first phase $V^{(7)}$ was not allowed to update, and we set $V^{(7)}_{t+1}$ equal to a random number. After 50,000 steps of training we allowed $V^{(7)}$ to update normally. Figure 1 (RHS) summarizes the results of our experiment. The results are averaged over 100 independent runs of the experiment, and window averaged with a window size of 1000.

The results highlight several simple but important points. During phase one, the error of the first six predictions (black line) quickly plateaus well above zero, because the predictions are inaccurate. During this phase the $V^{(7)}_{t+1}$ is random (red line), and thus the regularization scheme keeps the weights of all the features constructed from $V^{(7)}$ near zero (blue line). This prevents $V^{(7)}$'s inaccurate prediction from destabilizing the other predictions. In the second phase, we see $V^{(7)}$ learns quickly and it's influence on the other predictions rises (increase in blue line). Once the $V^{(7)}$ becomes reasonably accurate (but not yet zero error), it becomes a useful predictive feature and the error of the remaining predictions quickly reduces to zero. The use of a clipped regularizer sufficiently reduced bias and enabled prediction error to decrease to zero. The use of tile coding significantly increased the convergence rate in phase 2—a factor of three speedup.

Either form of stabilization—non-linear expansion through tile coding or regularization—were sufficient to learn in our experiment, though using both improved learning speed significantly. However, without the predictive features constructed from $V^{(7)}$, the other six predictions could never obtain reasonable accuracy. Our experiment provides a modest demonstration of one of the main ideas of continual learning: expanding the representation with predictive features is essential for efficient learning.

5. Future work

Our experiments demonstrate that our proposed architecture can improve the stability of prediction learning, but theoretical questions remain. The conventional notion of convergence is not suitable for continual learning where learning is never meant to end, and the agent should be tracking (Sutton et al., 2007). The regret formalism, could provide a mechanism for analyzing our proposed learning algorithms. Even this formalism, however, often considers a single best weighting across experts in hindsight, or does not consider the direct impact of this learned weighting on the experts (features). A new analysis paradigm may be needed to quantify sound learning for predictive representations, and more widely for continual learning.

2. Learning in cycle world can be also be achieved by adding a form of eligibility traces to the TD-net learning rule or using the recurrent neural network architecture of Silver (2012) .

References

B. Boots, S. Siddiqi, and G. Gordon. Closing the learning-planning loop with predictive state representations. *The International Journal of Robotics Research*, 2011.

M. H. Bowling, P. McCracken, M. James, J. Neufeld, and D. F. Wilkinson. Learning predictive state representations using non-blind policies. In *International Conference on Machine Learning*, 2006.

M. R. James and S. Singh. Learning and discovery of predictive state representations in dynamical systems with reset. In *International Conference on Machine Learning*, 2004.

M. L. Littman, R. S. Sutton, and S. Singh. Predictive representations of state. In *Advances in Neural Information Processing Systems*, 2001.

H. Maei. *Gradient Temporal-Difference Learning Algorithms*. PhD thesis, University of Alberta, 2011.

H. Maei, C. Szepesvári, S. Bhatnagar, D. Precup, D. Silver, and R. S. Sutton. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*, 2009.

S. Mahadevan, B. Liu, P. S. Thomas, W. Dabney, S. Giguere, N. Jacek, I. Gemp, and J. Liu. Proximal reinforcement learning: A new theory of sequential decision making in primal-dual spaces. *CoRR abs/1405.6757*, 2014.

P. McCracken and M. H. Bowling. Online discovery and learning of predictive state representations. In *Advances in Neural Information Processing Systems*, 2005.

E. J. Rafols. *Temporal abstraction in temporal-difference networks*. PhD thesis, University of Alberta, 2006.

M. Rosencrantz, G. Gordon, and S. Thrun. Learning low dimensional predictive representations. In *International Conference on Machine Learning*, 2004.

M. Rudary, S. Singh, and D. Wingate. Predictive linear-Gaussian models of stochastic dynamical systems. In *Conference on Uncertainty in Artificial Intelligence*, 2005.

D. Silver. Gradient Temporal Difference Networks. In *European Workshop on Reinforcement Learning*, 2012.

S. Singh, M. Littman, N. K. Jong, D. Pardoe, and P. Stone. Learning predictive state representations. In *International Conference on Machine Learning*, 2003.

R. S. Sutton, E. J. Rafols, and A. Koop. Temporal Abstraction in Temporal-difference Networks. In *Advances in Neural Information Processing Systems*, 2005.

R. S. Sutton, A. Koop, and D. Silver. On the role of tracking in stationary environments. In *Proceedings of the 24th international conference on Machine learning*, pages 871–878. ACM, 2007.

R. S. Sutton, H. Maei, D. Precup, and S. Bhatnagar. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning*, 2009a.

R. S. Sutton, C. Szepesvári, and H. Maei. A convergent O (n) algorithm for off-policy temporal-difference learning with linear function approximation. In *Advances in Neural Information Processing Systems*, 2009b.

R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. Pilarski, A. White, and D. Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *International Conference on Autonomous Agents and Multiagent Systems*, 2011.

B. Tanner and R. S. Sutton. Temporal-Difference Networks with History. In *International Joint Conference on Artificial Intelligence*, 2005.

A. White. *Developing a predictive approach to knowledge*. PhD thesis, University of Alberta, 2015.

E. Wiewiora. Learning predictive representations from a history. In *International Conference on Machine Learning*, 2005.

D. Wingate and S. Singh. Kernel predictive linear Gaussian models for nonlinear stochastic dynamical systems. In *International Conference on Machine Learning*, 2006.

B. Wolfe and S. P. Singh. Predictive state representations with options. In *International Conference on Machine Learning*, 2006.

B. Wolfe, M. R. James, and S. Singh. Learning predictive state representations in dynamical systems without reset. In *International Conference on Machine Learning*, 2005.

Y. Yu, X. Zheng, M. Marchetti-Bowick, and E. P. Xing. Minimizing Nonconvex Non-Separable Functions. In *International Conference on Artificial Intelligence and Statistics*, 2015.