Learning Retrospective Knowledge with Reverse Reinforcement Learning

Shangtong Zhang
University of Oxford

Vivek Veeriah
University of Michigan, Ann Arbor

Shimon Whiteson
University of Oxford

Abstract

We present a Reverse Reinforcement Learning (Reverse RL) approach for representing retrospective knowledge. General Value Functions (GVFs) have enjoyed great success in representing predictive knowledge, i.e., answering questions about possible future outcomes such as “how much fuel will be consumed in expectation if we drive from A to B?”. GVs, however, cannot answer questions like “how much fuel do we expect a car to have given it is at B at time t?”. To answer this question, we need to know when that car had a full tank and how that car came to B. Since such questions emphasize the influence of possible past events on the present, we refer to their answers as retrospective knowledge. In this paper, we show how to represent retrospective knowledge with Reverse GVs, which are trained via Reverse RL. We demonstrate empirically the utility of Reverse GVs in both representation learning and anomaly detection.

1 Introduction

Much knowledge can be formulated as answers to predictive questions (Sutton, 2009), for example, “to know that Joe is in the coffee room is to predict that you will see him if you went there” (Sutton, 2009). Such knowledge is referred to as predictive knowledge (Sutton, 2009; Sutton et al., 2011). General Value Functions (GVFs, Sutton et al., 2011) are commonly used to represent predictive knowledge. GVs are essentially the same as canonical value functions (Puterman, 2014; Sutton and Barto, 2018). However, the policy, the reward function, and the discount function associated with GVs are usually carefully designed such that the numerical value of a GVF at certain states matches the numerical answer to certain predictive questions. In this way, GVs can represent predictive knowledge.

Consider the concrete example in Figure 1 where a microdrone is doing a random walk. The microdrone is initialized somewhere with 100% battery. L4 is a power station where its battery is recharged to 100%. Each clockwise movement consumes 2% of the battery, and each counterclockwise movement consumes 1% (for simplicity, we assume negative battery levels, e.g., -10%, are legal). Furthermore, each movement fails with probability 1%, in which case the microdrone remains in the same location and no energy is consumed. An example of a predictive question in this system is:

**Question 1.** Starting from L1, how much energy will be consumed in expectation before the next charge?

To answer this question, we can model the system as a Markov Decision Process (MDP). The policy is uniformly random and the reward for each movement is the

*Correspondence to shangtong.zhang@cs.ox.ac.uk

additive inverse of the corresponding battery consumption. Whenever the microdrone reaches state L4, the episode terminates. Under this setup, the answer to Question 1 is the expected cumulative reward when starting from L1, i.e., the state value of L1. Hence, GVFs can represent the predictive knowledge in Question 1. As a GVF is essentially a value function, it can be trained with any data stream from agent-environment interaction via Reinforcement Learning (RL, Sutton and Barto 2018), demonstrating the generality of the GVF approach. Importantly, the most appealing feature of GVFs is their compatibility with off-policy learning, making this representation of predictive knowledge scalable and efficient. For example, in the Horde architecture (Sutton et al. 2011), many GVFs are learned in parallel with gradient-based off-policy temporal difference methods (Sutton et al. 2009b,a; Maei 2011). In the microdrone example, we can learn the answer to Question 1 under many different conditions (e.g., when the charging station is located at L2 or when the microdrone moves clockwise with probability 80%) simultaneously with off-policy learning by considering different reward functions, discount functions, and policies.

GVFs, however, cannot answer many other useful questions, e.g., if at some time $t$, we find the microdrone at L1, how much battery do we expect it to have? As such questions emphasize the influence of possible past events on the present, we refer to their answers as retrospective knowledge. Such retrospective knowledge is useful, for example, in anomaly detection. Suppose the microdrone runs for several weeks by itself while we are traveling. When we return at time $t$, we find the microdrone is at L1. We can then examine the battery level and see if it is similar to the expected battery at L1. If there is a large difference, it is likely that there is something wrong with the microdrone. There are, of course, many methods to perform such anomaly detection. For example, we could store the full running log of the microdrone during our travel and examine it when we are back. The memory requirement to store the full log, however, increases according to the length of our travel. By contrast, if we have retrospective knowledge, i.e., the expected battery level at each location, we can program the microdrone to log its battery level at each step (overwriting the record from the previous step). We can then examine the battery level when we are back and see if it matches our expectation. The current battery level can be easily computed via the previous battery level and the energy consumed at the last step, using only constant computation per step. The storage of the battery level requires only constant memory as we do not need to store the full history, which would not be feasible for a microdrone. Thus retrospective knowledge provides a memory-efficient way to perform anomaly detection. Of course, this approach may have lower accuracy than storing the full running log. This is indeed a trade-off between accuracy and memory, and we expect applications of this approach in memory-constrained scenarios such as embedded systems.

To know the expected battery level at L1 at time $t$ is essentially to answer the following question:

**Question 2.** How much energy do we expect the microdrone to have consumed since the last time it had 100% battery given that it is at L1 at time $t$?

Unfortunately, GVFs cannot represent retrospective knowledge (e.g., the answer to Question 2) easily. GVFs provide a mechanism to ignore all future events after reaching certain states via setting the discount function at those states to be 0. This mechanism is useful for representing predictive knowledge. For example, in Question 1, we do not care about events after the next charge. For retrospective knowledge, we, however, need a mechanism to ignore all previous events before reaching certain states. For example, in Question 2, we do not care about events before the last time the microdrone had 100% battery. Unfortunately, GVFs do not have such a mechanism. In Appendix A, we describe several tricks that attempt to represent retrospective knowledge with GVFs and explain why they are invalid.

In this paper, we propose Reverse GVFs to represent retrospective knowledge. Using the same MDP formulation of the microdrone system, let the random variable $G_t$ denote the energy the microdrone has consumed at time $t$ since the last time it had 100% battery. To answer Question 2, we are interested in the conditional expectation of $G_t$ given that $S_t = L1$. We refer to functions describing such conditional expectations as Reverse GVFs, which we propose to learn via Reverse Reinforcement Learning. The key idea of Reverse RL is still bootstrapping, but in the reverse direction. It is easy to see that $G_t$ depends on $G_{t-1}$ and the energy consumption from $t-1$ to $t$. In general, the quantity of interest at time $t$ depends on that at time $t-1$ in Reverse RL. This idea of bootstrapping from the past has been explored by Wang et al. (2007, 2008), Hallak and Mannor (2017), Gelada and Bellemare (2019), Zhang et al. (2020a) but was limited to the density ratio learning setting. We propose several Reverse RL algorithms and prove their convergence under linear function approximation. We also propose Distributional Reverse RL algorithms akin to Distributional
We consider an infinite-horizon Markov Decision Process (MDP) with a finite state space \( S \), a finite action space \( A \), a transition kernel \( p: S \times S \times A \rightarrow [0, 1] \), and an initial distribution \( \mu_0: S \rightarrow [0, 1] \). In the GVF framework, users define a reward function \( r: S \times A \rightarrow \mathbb{R} \), a discount function \( \gamma: S \rightarrow [0, 1] \), and a policy \( \pi: A \times S \rightarrow [0, 1] \) to represent certain predictive questions. An agent is initialized at \( S_0 \) according to \( \mu_0 \). At time step \( t \), an agent at a state \( S_t \) selects an action \( A_t \) according to \( \pi(\cdot|S_t) \), receives a bounded reward \( R_{t+1} \) satisfying \( \mathbb{E}[R_{t+1}] = r(S_t, A_t) \), and proceeds to the next state \( S_{t+1} \) according to \( p(\cdot|S_t, A_t) \). We then define the return at time step \( t \) recursively as:

\[
G_t = R_{t+1} + \gamma(S_{t+1})G_{t+1},
\]

which allows us to define the general value function \( v_\pi(s) \equiv \mathbb{E}[G_t|S_t = s] \). The general value function \( v_\pi \) is essentially the same as the canonical value function \( \mathbb{V}^\pi = \mathbb{E}[\sum_{t'=0}^{T-1} \gamma^t r(s_t, a_t)|S_0 = s_0] \) (Puterman, 2014; Sutton and Barto, 2018). The name “general” emphasizes its usage in representing predictive knowledge. In the literature, \( G_t \) is often termed the GVF.
see that \( \bar{G}_2 \) is exactly the energy the microdrone has consumed since its last charge. In general, if we find the microdrone at \( L_1 \) at time \( t \), the expectation of the energy that the microdrone has consumed since its last charge is exactly \( \mathbb{E}_{\pi,p,t}[G_t | S_t = L_1] \). Note the answer to Question 2 is not homogeneous in \( t \). For example, suppose the microdrone is initialized at \( L_4 \) at \( t = 0 \). If we find it at \( L_1 \) at \( t = 1 \), it is trivial to see the microdrone has consumed 2% battery. By contrast, if we find it at \( L_1 \) at \( t = 100 \), computing the energy consumption since the last time it had 100% battery is nontrivial. It is inconvenient that the answer depends the time step \( t \) but fortunately, we can show the following:

**Assumption 1.** The chain induced by \( \pi \) is ergodic and \( (I - P^T \Gamma)^{-1} \) exists.

**Theorem 1.** Under Assumption 1, the limit \( \lim_{t \to \infty} \mathbb{E}[\bar{G}_t | S_t = s] \) exists, which we refer to as \( \bar{v}_\pi(s) \). Furthermore, we define the reverse Bellman operator \( \bar{T} \) as

\[
\bar{T} y = D^{-1}_\pi \bar{P}^T_\pi \bar{D}_\pi r + D^{-1}_\pi P^T_\pi \Gamma D y,
\]

where \( D_\pi = \text{diag}(d_\pi) \in \mathbb{R}^{|S| \times |S|} \) with \( d_\pi \) being the stationary distribution of the chain induced by \( \pi \), \( \bar{P}_\pi \in \mathbb{R}^{(|S| \times |A| \times |S|)} \) is the transition matrix, i.e., \( \bar{P}_\pi((s', a), s') = p(s' | s, a) \), and \( D_\pi = \text{diag}(d_\pi) \in \mathbb{R}^{(|S| \times |A| \times |S|)} \) with \( d_\pi(s, a) = d_\pi(s) \pi(a | s) \). Then \( \bar{T} \) is a contraction mapping w.r.t. some weighted maximum norm, and \( \bar{v}_\pi \) is its unique fixed point. We have \( \bar{v}_\pi = D^{-1}_\pi (I - P^T \Gamma)^{-1} \bar{P}^T_\pi \bar{D}_\pi r \).

Assumption 1 can be easily fulfilled in the real world as long as the problem we consider has a recurring structure. The proof of Theorem 1 is based on [Sutton et al. (2016); Zhang et al. (2019, 2020d)] and is detailed in the appendix. Theorem 1 states that the numerical value of \( \bar{v}_\pi \) (L1) approximately answers Question 2. When Question 2 is asked for a large enough \( t \), the error in the answer \( \bar{v}_\pi \) is arbitrarily small. We call \( \bar{v}_\pi(s) \) a Reverse General Value Function, which approximately encodes the retrospective knowledge, i.e., the answer to the retrospective question induced by \( \pi \), \( r \), \( \gamma \), \( t \) and \( s \).

Based on the reverse Bellman operator \( \bar{T} \), we now present the Reverse TD algorithm. Let us consider linear function approximation with a feature function \( x \). We use the reverse Bellman operator \( \bar{T} \) and is detailed in the appendix. Theorem 1 states that the numerical value of \( \bar{v}_\pi \) (L1) approximately answers Question 2. When Question 2 is asked for a large enough \( t \), the error in the answer \( \bar{v}_\pi \) is arbitrarily small. We call \( \bar{v}_\pi(s) \) a Reverse General Value Function, which approximately encodes the retrospective knowledge, i.e., the answer to the retrospective question induced by \( \pi \), \( r \), \( \gamma \), \( t \) and \( s \).

**Proposition 1.** (Convergence of Reverse TD) Under Assumption 1, assuming \( X \) has linearly independent columns, then the iterate \( \{w_t\} \) generated by Reverse TD (Eq 1) satisfies \( \lim_{t \to \infty} w_t = -A^{-1}b \) with probability 1, where \( A \) is the transpose of \( P_\Gamma \Gamma - I \) times \( D_x X \), \( b \) is \( X^\top \bar{P}^T_\pi \bar{D}_\pi r \).

The proof of Proposition 1 is based on the proof of the convergence of linear TD in [Bertsekas and Tsitsiklis (1996)]. In particular, we need to show that \( A \) is negative definite. Details are provided in the appendix. For a sanity check, it is easy to verify that in the tabular setting (i.e., \( X = I \)), \( -A^{-1}b = \bar{v}_\pi \) indeed holds. Inspired by the success of TD(\( \lambda \)) (Sutton (1988) and COP-TD(\( \lambda \)) (Hallak and Mannor 2017), we also extend Reverse TD to Reverse TD(\( \lambda \)), which updates \( w_{t+1} \) as

\[
w_{t+1} = w_t + \alpha_t (R_t + \gamma(S_{t-1})x_{t-1}^\top w_t - x_t^\top w_t)x_t,
\]

where \( x_t \) is shorthand, and \( \{\alpha_t\} \) is a deterministic positive nonincreasing sequence satisfying the Robbins-Monro condition (Robbins and Monro 1951), i.e., \( \sum_t \alpha_t = \infty \), \( \sum_t \alpha_t^2 < \infty \). We have

\[
\lim_{t \to \infty} w_t = \bar{v}_\pi - A^{-1}b.
\]

With \( \lambda = 1 \), Reverse TD(\( \lambda \)) reduces to supervised learning.

**Distributional Learning.** In anomaly detection with Reverse GVs, we compare the observed quantity (a scalar) with our retrospective knowledge (a scalar, the conditional expectation). It is not clear how to translate the difference between the two scalars into a decision about whether there is an anomaly. If our retrospective knowledge is a distribution instead, we can perform anomaly detection from a probabilistic perspective. To this end, we propose Distributional Reverse TD, akin to [Bellemare et al. (2017); Rowland et al. (2018)].

We use \( \eta_t^s \in \mathcal{P}(\mathbb{R}) \) to denote the conditional probability distribution of \( G_t \), given \( S_t = s \), where \( \mathcal{P}(\mathbb{R}) \) is the set of all probability measures over the measurable space \((\mathbb{R}, \mathcal{B}(\mathbb{R}))\), with \( \mathcal{B}(\mathbb{R}) \) being the Borel sets of \( \mathbb{R} \). Moreover, we use \( \eta_t^s \in \mathcal{P}(\mathbb{R})^{|S|} \) to denote the vector whose \( s \)-th element is \( \eta_t^s \). By the definition of \( G_t \), we have for any \( E \in \mathcal{B}(\mathbb{R}) \)

\[
\eta_t^s(E) = \int_{\mathbb{R} \times S} (f_{r,s} \# \eta_t^{s-1})(E) d \Pr(S_{t-1} = s, R_t = r | S_t = s),
\]

(2)
where \( f_{r,\bar{s}} : \mathbb{R} \rightarrow \mathbb{R} \) is defined as \( f_{r,\bar{s}}(x) = r + \gamma(\bar{s})x \), and \( f_{r,\bar{s}} \# \eta_{t-1} = \mathcal{B}(\mathbb{R}) \rightarrow [0,1] \) is the push-forward measure, i.e., \( (f_{r,\bar{s}} \# \eta_{t-1})(E) = \int_{\mathbb{R}} \mathcal{B}(\mathbb{R}) \rightarrow [0,1] \) is the preimage of \( E \). To study \( \eta_t^\theta \) when \( t \to \infty \), we define

\[
\rho(s, r | s) = \lim_{t \to \infty} \Pr(S_t = s, R_t = r | S_{t-1} = s) = \frac{d_{\eta_t}^x}{d_{\eta_t}^s} \sum \pi(\bar{a}|\bar{s})p(s, \bar{a}) \Pr(r | s, \bar{a}).
\]

When \( t \to \infty \), Eq (3) suggests \( \eta_t^\theta \) evolves according to \( \eta_t^\theta = \int_{\mathbb{R} \times \mathcal{S}} (f_{r,\bar{s}} \# \eta_{t-1})(E) d \rho(s, r | s) \).

We, therefore, define the distributional reverse Bellman operator \( \mathcal{T} : (\mathcal{P}([\mathbb{R}])^{|\mathcal{S}|}) \rightarrow (\mathcal{P}([\mathbb{R}])^{|\mathcal{S}|}) \) as

\[
\mathcal{T} \eta^\theta = \int_{\mathbb{R}_\mathcal{S}} (f_{r,\bar{s}} \# \eta^\theta) d \rho(s, r | s).
\]

We have

**Proposition 2.** Under Assumption 1, \( \mathcal{T} \) is a contraction mapping w.r.t. a metric \( d \), and we refer to its fixed point as \( \eta^\star \). Assuming \( \mu_0 = d_{\eta^\star} \), then \( \lim_{t \to \infty} d(\eta_t, \eta^\star) = 0 \).

We now provide a practical algorithm to approximate \( \eta^\star \) based on quantile regression, akin to Dabney et al. (2017). We use \( N \) quantiles with quantile levels \( \{\tau_i\}_{i=1}^N \), where \( \tau_i = \frac{(i-1)/N+1/N}{2} \). The measure \( \eta^\star \) is approximated with \( \frac{1}{N} \sum_{i=1}^N \delta_{\eta_i(s, \theta)} \), where \( \delta_z \) is a Dirac at \( z \). \( q_i(s; \theta) \) is a quantile corresponding to the quantile level \( \tau_i \), and \( \theta \) is learnable parameters. Given a transition \((s, a, r, s')\), we train \( \theta \) to minimize the following quantile regression loss

\[
L(\theta) = \sum_{i=1}^N \rho_{\tau_i}(r + \frac{\tau_i(s)}{N} \sum_{k=1}^N q_i(s; \theta) - \frac{1}{N} \sum_{k=1}^N q_i(s'; \theta)),
\]

where \( \theta \) contains the parameters of the target network (Mnih et al., 2015), which is synchronized with \( \theta \) periodically, and \( \rho_{\tau_i}(x) = |\tau_i - \mathbb{E}_s q_i(x)| \) is the quantile regression loss function. \( \mathcal{H}_\kappa(x) \) is the Huber loss, i.e., \( \mathcal{H}_\kappa(x) = 0.5x^2 \mathbb{I}_{x<\kappa} + \kappa(|x| - 0.5\kappa) \mathbb{I}_{x>\kappa} \), where \( \kappa \) is a hyperparameter. Dabney et al. (2017) provide more details about quantile-regression-based distributional RL.

**Off-policy Learning.** We would also like to be able to answer to Question 2 without making the microdrone do a random walk, i.e., we may have another policy \( \mu \) for the microdrone to collect data. In this scenario, we want to learn \( \bar{v}_\pi \) off-policy. We consider Off-policy Reverse TD, which updates \( w_t \) as:

\[
w_{t+1} = w_t + \alpha_t \tau(S_{t-1}) \rho(S_{t-1}, A_{t-1}) (R_t + \gamma(S_{t-1})x_{t-1}^T w_t - x_{t-1}^T w_t)x_t,
\]

where \( \tau(s) = \frac{d_{\eta_t}^s}{d_{\eta_t}^s} \rho(s, a) \cdot \frac{\pi(a | s)}{\mu(a | s)} \) and \( \{S_0, A_0, R_1, S_1, \ldots\} \) is obtained by following the behavior policy \( \mu \). Here we assume access to the density ratio \( \tau(s) \), which can be learned via Hallak and Mannor (2017), Gelada and Bellemare (2019a); Nachum et al. (2019), Zhang et al. (2020a).

**Proposition 3.** (Convergence of Off-policy Reverse TD) Under Assumption 1, assuming \( X \) has linearly independent columns, and the chain induced by \( \mu \) is ergodic, then the iterate \( \{w_t\} \) generated by Off-policy Reverse TD (Eq (3)) satisfies \( \lim_{t \to \infty} w_t = -A^{-1}b \) with probability 1.

Off-policy Reverse TD converges to the same point as on-policy Reverse TD. This convergence relies heavily on having the true density ratio \( \tau(s) \). When using a learned estimate for the density ratio, approximate inversion is impossible and thus convergence is not ensured. It is straightforward to consider a GTD (Sutton et al., 2009b, Mael, 2011) analogue, Reverse GTD, similar to Gradient Emphasis Learning in Zhang et al. (2020d). The convergence of Off-Policy Reverse GTD is straightforward (Zhang et al., 2020d), but is a different point from On-policy Reverse TD.

**4 Experiments**

**The Effect of \( \lambda \).** At time step \( t \), the reverse return \( \bar{G}_t \) is known and can approximately serve as a sample for \( \bar{v}_\pi(S_t) \). It is natural to model this as a regression task where the input is \( S_t \), and the target is \( \bar{G}_t \). This is indeed Reverse TD(1). So we first study the effect of \( \lambda \) in Reverse TD(\( \lambda \)). We consider the microdrone example in Figure 1. The dynamics are specified in Section 1. The reward function and the discount function are specified in Section 2. The policy \( \pi \) is uniformly random. We use a tabular representation and compute the ground truth \( \bar{v}_\pi \) analytically. We vary \( \lambda \) in \{0, 0.3, 0.7, 0.9, 1.0\}. For each \( \lambda \), we use a constant step size \( \alpha \) tuned from \{0.5 \times 10^{-3}, 10^{-2}, 5 \times 10^{-2}\}. We report the Mean Value Error (MVE) against training steps in Figure 2. At a time step \( t \), assuming our estimation is \( \bar{V} \), the MVE is computed as \( ||\bar{V} - \bar{v}_\pi||_2^2 \). The results show that the bias of the estimate decreases quickly at the beginning. As a result, variance of the update target becomes the major obstacle in the learning process, which explains why the best performance is achieved by smaller \( \lambda \) in this experiment.
Anomaly Detection.

Tabular Representation. Consider the microdrone example once again (Figure 1). Suppose we want the microdrone to follow a policy \( \pi \) where \( \pi(a_1|s) = 0.1 \forall s \). However, something can go wrong when the microdrone is following \( \pi \). For example, it may start to take \( a_1 \) with probability 0.9 at all states due to a malfunctioning navigation system, which we refer to as a policy anomaly. The microdrone may also consume 2% extra battery per step with probability 0.5 due to a malfunctioning engine, which we refer to as a reward anomaly, i.e., the reward \( R_t \) becomes \( R_t + 2 \) with probability 0.5. We cannot afford to monitor the microdrone every time step but can do so occasionally, and we hope if something has gone wrong we can discover it. Since it is a microdrone, it does not have the memory to store all the logs between examinations. We now demonstrate that Reverse GVFs can discover such anomalies using only constant memory and computation.

Our experiment consists of two phases. In the first phase, we train Reverse GVFs off-policy. Our behavior policy \( \mu \) is uniformly random with \( \mu(a_1|s) = 0.5 \forall s \). The target policy is \( \pi \) with \( \pi(a_1|s) = 0.1 \forall s \). Given a transition \((s, a, r, s')\) following \( \mu \), we update the parameters \( \theta \), which is a look-up table in this experiment, to minimize \( L(\theta) \). In this way, we approximate \( \eta^\pi_t \) with \( N = 20 \) quantiles for all \( s \). The MVE against training steps is reported in Figure 2.

In the second phase, we use the learned \( \eta^\pi_t \) from the first phase for anomaly detection when we actually deploy \( \pi \). Namely, we let the microdrone follow \( \pi \) for \( 2 \times 10^4 \) steps and compute \( G_t \) on the fly. In the first \( 10^4 \) steps, there is no anomaly. In the second \( 10^4 \) steps, the aforementioned reward anomaly or policy anomaly happens every step. We aim to discover the anomaly from the information provided by \( G_t \) and \( \eta^\pi_t \). Namely, we report the probability of anomaly as

\[
\text{prob}_{\text{anomaly}}(G_t) = 1 - \eta^{\text{Mix}}_t(\bar{G}_t - \Delta, \bar{G}_t + \Delta),
\]

where \( \Delta \) is a hyperparameter. If a larger \( \Delta \) is used, the reported probability of anomaly will in general be closer to 0, but then the algorithm becomes less sensitive to anomaly (e.g., if \( \Delta = \infty \), the output will always be 0). So \( \Delta \) achieves a trade-off between reducing the false alarms (i.e., making the output as low as possible when no anomaly) and increasing sensitivity to the anomaly. This approach for computing the probability of anomaly is simple but intuitive. A more formal approach requires properly defined priors over \( G_t \) and the occurrence of anomalies to make use of Bayes’ rule. However, those priors depend heavily on the application and complicate the presentation of the central idea to conduct anomaly detection with reverse GVF. We, therefore, use this simple approach in our paper. We believe detecting anomaly using only a single observation based on a known p.d.f. itself is an interesting statistical problem that is out of the scope of this paper. We use \( \Delta = 1 \) in our experiments. Moreover, we do not have access to \( \eta^{N_t}_t \) but only \( N \) estimated quantiles \( \{q_i(S_t; \theta)\}_{i=1,...,N} \). To compute \( \text{prob}_{\text{anomaly}}(G_t) \), we need to first find a distribution whose quantiles are \( q_i(S_t; \theta) \). This operation is referred to as imputation in [Rowland et al., 2018]. Such a distribution is not unique. The commonly used imputation strategy for quantile-regression-based distributional RL is \( \frac{1}{N} \sum_{i=1}^{N} \delta_{q_i(S_t; \theta)} \) [Dabney et al., 2017]. This distribution, however, makes it difficult to compute \( \text{prob}_{\text{anomaly}}(G_t) \). Inspired by the fact that a Dirac can be regarded as the limit of a normal distribution with a decreasing standard derivation, we define our approximation for \( \eta^{N_t}_t \) as \( \tilde{\eta}^{N_t}_t = \frac{1}{N} \sum_{i=1}^{N} \mathcal{N}(q_i(S_t; \theta), \sigma^2) \), where \( \sigma \) is a hyperparameter and we use \( \sigma = 1 \) in our experiments. Note \( \tilde{\eta}^{N_t}_t \) does not necessarily have the quantiles \( q_i(S_t; \theta) \). We report \( 1 - \tilde{\eta}^{N_t}_t(\bar{G}_t - \Delta, \bar{G}_t + \Delta) \) against time steps in Figure 3. When the anomaly occurs after the first \( 10^4 \) steps, the probability of anomaly reported by Reverse GVF becomes high.

Non-linear Function approximation. We now consider Reacher from OpenAI gym [Brockman et al., 2016] and use neural networks as a function approximator for \( q_i(s; \theta) \). Our setup is the same

as the tabular setting except that the tasks are different. For a state $s$, we define $\gamma(s) = 0$ if the distance between the end of the robot arm and the target is less than 0.02. Otherwise we always have $\gamma(s) = 1$. When the robot arm reaches a state $s$ with $\gamma(s) = 0$, the arm and the target are reinitialized randomly. We first train a deterministic policy $\mu_d$ with TD3 [Fujimoto et al. 2018] achieving an average episodic return of $-4$. In the first phase, we use a Gaussian behavior policy $\mu(s) \sim \mathcal{N}(\mu_d(s), 0.5^2)$. The target policy is $\pi(s) \sim \mathcal{N}(\mu_d(s), 0.1^2)$. In the second phase, we consider two kinds of anomaly. In the policy anomaly, we consider three settings where the policy $\pi(s)$ becomes $\mathcal{N}(\mu_d(s), 0.9^2), \mathcal{N}(\mu_d(s), 1.8^2)$, and $\mathcal{N}(\mu_d(s), 2.7^2)$ respectively. In the reward anomaly, we consider three settings where with probability 0.5 the reward $R_t$ becomes $R_t = 1$, $R_t = -5$, and $R_t = -10$ respectively. We report the estimated probability of anomaly in Figure 3. When an anomaly happens after the first $10^4$ steps, the probability of anomaly reported by Reverse GVF becomes high. In Figure 3, the probability of anomaly is higher than 0.8. This is mainly due to the larger variance of the observed reward (compared with the toy MDP used in Figure 3), resulting from the large stochasticity of the policy being followed. When the variance of a random variable is large, the probability mass is not concentrated. Consequently, the information that a single observation can provide is less. So our anomaly detection has a higher chance for a false alarm.

We note that the goal of this work is not to achieve a new state-of-the-art in anomaly detection. Simple heuristics are enough to outperform our approach in the tested domains. Instead, we want to highlight the potential of Reverse-RL-based anomaly detection. An agent can obtain and maintain a huge amount of retrospective knowledge easily via off-policy Reverse RL with function approximation. Given the learned retrospective knowledge, anomaly detection can be simple and cheap. Our empirical study aims to provide a proof-of-concept of this new paradigm. There are indeed open questions in this new paradigm, e.g., the possible large variance of $G_t$ and the threshold for anomaly alert, which we leave for future work.

**Representation Learning.** Veeriah et al. (2019) show that automatically discovered GVFs can be used as auxiliary tasks (Jaderberg et al. 2016) to improve representation learning, yielding a performance boost in the main task. Let $r$ and $\gamma$ be the reward function and the discount factor of the main task. Veeriah et al. (2019) propose two networks for solving the main task: a main task
We now show that automatically discovered Reverse GVFs can also be used as auxiliary tasks at Atari games. Gelada and Bellemare (2019) propose stochastic approximation algorithms (discounted) and emulate the followon trace perspective is one origin of bootstrapping in the reverse direction, and the reverse return $G_t$ is inspired by the followon trace $F_i$ in Sutton et al. (2016), which is defined as $F_i = i(S_i) + \gamma(S_i)\rho_{t-1}F_{i-1}$, where $i : S \rightarrow [0, \infty)$ is a user-defined interest function specifying user’s preference for different states. Sutton et al. (2016) use the followon trace to reweight value function update in Emphatic TD. Later on, Zhang et al. (2020d) propose to learn the conditional expectation $\lim_{t \rightarrow \infty} \mathbb{E}[F_i | S_t = s]$ with function approximation in off-policy actor-critic algorithms. This followon trace perspective is one origin of bootstrapping in the reverse direction, and the followon trace is used only for stabilizing off-policy learning. The second origin is related to learning the stationary distribution of a policy, which dates back to Wang et al. (2007, 2008) in dual dynamic programming for stable policy evaluation and policy improvement. Later on, Hallak and Mannor (2017), Gelada and Bellemare (2019) propose stochastic approximation algorithms (discounted) COP-TD to learn the density ratio, i.e. the ratio between the stationary distribution of the target policy and that of the behavior policy, to stabilize off-policy learning. Our Reverse TD differs from the discounted COP-TD in that (1) Reverse TD is on-policy and does not have importance sampling ratios, while discounted COP-TD is designed only for off-policy setting, as there is no density ratio in the on-policy setting. (2) Reverse TD uses $R_i$ in the update, while discounted COP-TD uses a carefully designed constant. The third origin is an application of RL in web page ranking (Yao and Schuurmans 2013), where a different reverse Bellman equation is proposed to learn the authority score function. Although the idea of reverse bootstrapping is not new, we want to highlight that this paper is the first to apply this idea for representing retrospective knowledge and show its utility in 

4Those ten Atari games are the ten where the IMPALA+PixelControl agent achieves the largest improvement over the plain IMPALA agent over all 57 Atari games (Veeriah et al. 2019).
Anomaly detection has been widely studied in the machine learning community (e.g., see Chandola et al. (2009, 2010); Chalapathy and Chawla (2019)). Another approach for representing retrospective knowledge is to work directly with a reversed chain. Using Reverse RL for anomaly detection, however, Jaderberg et al. (2016) use reward prediction and pixel control as auxiliary tasks. Distributional RL provides a probabilistic perspective for anomaly detection. So misjudgment is possible. Reverse-RL makes it possible to implement anomaly detection with little extra memory. This is particularly important for embedded systems with limited memory, e.g., satellites, spacecrafts, microdrones, and IoT devices. The saved memory can be used to improve other functionalities of those systems. Systems where memory is not a bottleneck, e.g., self-driving cars, benefit from Reverse-RL-based anomaly detection as well, as saving memory saves energy, making them more environment-friendly.

Reverse-RL-based anomaly detection as well, as saving memory saves energy, making them more environment-friendly.

Users may have to make a decision considering other available information as well to reach a

6 Conclusion

In this paper, we present Reverse GVF for representing retrospective knowledge and formalize the Reverse RL framework. We demonstrate the utility of Reverse GVFs in both anomaly detection and representation learning. Investigating Reverse-GVF-based anomaly detection with real world data and applying Reverse GVFs in web page ranking are possible directions for future work.

Broader Impact

Reverse-RL makes it possible to implement anomaly detection with little extra memory. This is particularly important for embedded systems with limited memory, e.g., satellites, spacecrafts, microdrones, and IoT devices. The saved memory can be used to improve other functionalities of those systems. Systems where memory is not a bottleneck, e.g., self-driving cars, benefit from Reverse-RL-based anomaly detection as well, as saving memory saves energy, making them more environment-friendly.

Reverse-RL provides a probabilistic perspective for anomaly detection. So misjudgment is possible. Users may have to make a decision considering other available information as well to reach a
certain confidence level. Like any other neural network application, combining neural network with Reverse-RL-based anomaly detection is also vulnerable to adversarial attacks. This means the users, e.g., companies or governments, should take extra care for such attacks when making a decision on whether there is an anomaly or not. Otherwise, they may suffer from property losses. Although Reverse-RL itself does not have any bias or unfairness, if the simulator used to train reverse GVF is biased or unfair, the learned GVF are likely to inherit those bias or unfairness. Although Reverse-RL itself does not raise any privacy issue, to make a better simulator for training, users may be tempted to exploit personal data. Like any artificial intelligence system, Reverse-RL-based anomaly detection has the potential to greatly improve human productivity. However, it may also reduce the need for human workers, resulting in job losses.

Acknowledgments and Disclosure of Funding

SZ is generously funded by the Engineering and Physical Sciences Research Council (EPSRC). This project has received funding from the European Research Council under the European Union’s Horizon 2020 research and innovation programme (grant agreement number 637713). The experiments were made possible by a generous equipment grant from NVIDIA.

References


A  Failure in Representing Retrospective Knowledge with GVFs

One may consider answering Question\ref{question:2} with GVF via setting L4 to be the initial state and terminating an episode when the microdrone gets to L1. Then the value of L4 seems to be the answer to Question\ref{question:2}. To understand how this approach fails, let us consider transitions L4 \rightarrow L3 \rightarrow L4 \rightarrow L1. It then becomes clear that we are unable to design a Markovian reward for the transition L3 \rightarrow L4. This reward has to be non-Markovian to cancel all previously accumulated rewards. To make the reward Markovian, one may augment the state space with the battery level, which significantly increases the size of the state space. More importantly, this renders off-policy learning infeasible. The transition kernel on this augmented state space depends on the original reward function. So we cannot use off-policy learning to learn a GVF associated with a different reward function, as changing the reward function changes the transition kernel on the augmented state space. We can, of course, include the information about the new reward function into the augmented space. This, however, indicates the size of the state space grows exponentially with the number of reward functions we want to consider in off-policy learning. There is even a deeper defect. Let us consider the setting where we have two charging stations, say L2 and L4. Then if we want to use GVF directly as aforementioned assuming the aforementioned issues could somehow be solved, we need to set the initial state to L2 and L4 respectively. We then solve the two MDPs and compute \( v(L2) \) and \( v(L4) \) respectively. Finally, we may need to compute \( d_v(L2)v(L2) + d_v(L4)v(L4) \) as the answer, where \( d_v \) is the stationary distribution of the original MDP, which is, unfortunately, unknown. To summarize, there may be some retrospective knowledge that GVF can represent if enough tweaks are applied. But in general, representing retrospective with GVF suffers from poor generality and poor scalability.

B  Proofs

Lemma 1. (Corollary 6.1 in page 150 of Bertsekas and Tsitsiklis\cite{Bertsekas1989}) If \( Y \) is a square nonnegative matrix and \( \rho(Y) < 1 \), then there exists some vector \( w > 0 \) such that \( \|Y\|_{\infty} < 1 \). Here \( > \) is elementwise greater and \( \rho(\cdot) \) is the spectral radius. For a vector \( y \), its \( w \)-weighted maximum norm is \( \|y\|_w^\infty = \max_i \frac{y_i}{w_i} \). For a matrix \( Y \), \( \|Y\|_w^\infty = \max_{y \neq 0} \frac{\|Yy\|_w^\infty}{\|y\|_w^\infty} \).

B.1 Proof of Theorem\ref{theorem:1}

Proof. Given the similarity between \( \tilde{G}_t \) and the followon trace \( F_t \) as discussed in Section 5, the existence of \( \lim_{t \to \infty} \mathbb{E}_{\pi,p,r}[\tilde{G}_t|S_t = s] \) can be established in exactly the same way as\cite{Zhang2019} establish the existence of \( \lim_{t \to \infty} \mathbb{E}[F_t|S_t = s] \) in their Lemma 1. We therefore omit this to avoid verbatim repetition. We have

\[
\tilde{v}_\pi(s) = \lim_{t \to \infty} \mathbb{E}[\tilde{G}_t|S_t = s] = \lim_{t \to \infty} \mathbb{E}[R_t + \gamma(S_{t-1})\tilde{G}_{t-1}|S_t = s] = \lim_{t \to \infty} \sum_{\bar{s},\bar{a}} \Pr(S_{t-1} = \bar{s}, A_{t-1} = \bar{a}|S_t = s) \mathbb{E}[R_t + \gamma(S_{t-1})\tilde{G}_{t-1}|S_{t-1} = \bar{s}, A_{t-1} = \bar{a}] 
\]

(Law of total expectation)

\[
= \sum_{\bar{s},\bar{a}} \frac{d_\pi(s)\pi(\bar{a}|\bar{s})p(s,\bar{s},\bar{a})}{d_\pi(s)} (r(\bar{s},\bar{a}) + \gamma(\bar{s})\tilde{v}_\pi(\bar{s})) = \tilde{v}_\pi(s) 
\]

(Bayes’ rule) \hspace{0.5cm}(4)

The matrix form of Eq \ref{eq:4} is exactly \( \tilde{\pi} = D^{-1}_\pi P^T \tilde{D} \pi + D^{-1}_\pi P^T \Gamma D \pi \tilde{\pi} \), solving which leads to \( \tilde{\pi} = D^{-1}_\pi (I - D^T \pi \Gamma)^{-1} P^T \Gamma D \pi \tilde{\pi} \). Assumption\ref{assumption:1} implies \( \rho(P^T \Gamma) < 1 \). As \( Y_1 Y_2 \) and \( Y_2 Y_1 \) have the same eigenvalues (e.g., see Theorem 1.3.22 in\cite{Horn2012}), we have \( \rho(D^{-1}_\pi P^T \Gamma D \pi) = \rho(P^T \Gamma D \pi D^{-1}_\pi) < 1 \). Lemma\ref{lemma:1} then implies \( T \) is a contraction mapping w.r.t. some weighted maximum norm.

B.2 Proof of Proposition\ref{proposition:1}

We first state a lemma about the convergence of the following iterates

\[
w_{t+1} = w_t + \alpha_t (A(Y_t)w_t + b(Y_t))
\]
where \( \{Y_t\} \) is a Markov chain evolving in \( \mathcal{Y}, w_t \in \mathbb{R}^K, A : \mathcal{Y} \to \mathbb{R}^{K \times K}, b : \mathcal{Y} \to \mathbb{R}^K \).

**Assumption 2.** (Assumption 4.5 in Bertsekas and Tsitsiklis (1996))
(a) The step sizes \( \alpha_t \) are nonnegative, deterministic, and satisfy \( \sum \alpha_t = \infty, \sum_t \alpha_t^2 < \infty \).
(b) The chain \( \{Y_t\} \) has a stationary distribution \( \pi \).
(c) The matrix \( \bar{A} = \mathbb{E}_{y \sim p_y} [A(y)] \) is negative definite.
(d) There is a constant \( C_0 \) such that \( \|A(y)\| \leq C_0 \) and \( \|b(y)\| \leq C_0 \).
(e) There exists scalars \( 0 < C_1, 0 < \rho < 1 \) such that
\[
\|\mathbb{E}[A(Y_t)] - \bar{A}\| \leq C_1 \rho^t, \quad \|\mathbb{E}[b(Y_t)] - \bar{b}\| \leq C_1 \rho^t,
\]
where \( \bar{b} = \mathbb{E}_{y \sim p_y} [b(y)] \).

**Lemma 2.** (Proposition 4.8 in Bertsekas and Tsitsiklis (1996))
Under Assumption 2, \( \lim_{t \to \infty} \tilde{w}_t = -A^{-1}b \) with probability 1.

We now prove Theorem 1 via verifying Assumption 2 thus invoking Lemma 2.

**Proof.** We first consider a deterministic reward setting, i.e., we assume \( R_{t+1} = r(S_{t}, A_t) \). The Reverse TD update Eq (1) can be rearranged as
\[
w_{t+1} = w_t + \alpha_t (A(Y_t) w_t + b(Y_t)),
\]
where \( Y_t = (X_{t-1}, A_{t-1}, X_t), y = (s, a, s'), A(y) = \gamma(s)x(s')x(s)^\top - x(s)x(s')^\top, b(y) = r(s, a)x(s') \). It is easy to verify that \( \bar{A} \) is symmetric. Assumption 2(a) is satisfied automatically. Obviously \( \{Y_t\} \) is ergodic and its stationary distribution is \( p_y \) such that \( p_y = p_y p(a|s)p(s'|s, a) \). Assumption 2(b) is now satisfied.

We now verify Assumption 2(c). Our proof is inspired by the proof of Lemma 6.4 in Bertsekas and Tsitsiklis (1996). Let \( z \in \mathbb{R}^{|S|} / \{0\} \), we aim to show \( z^\top \bar{A} z < 0 \). As \( X \) has linearly independent columns, it suffices to show \( z^\top D_{\pi} z < 0 \). We have
\[
\|\Pi_{\pi} z\|_{D_{\pi}}^2 = \sum_s d_{\pi}(s) \gamma(s)^2 \left( \sum_{s'} P_{\pi}(s, s') z(s') \right)^2 \leq \sum_s d_{\pi}(s) \gamma(s) \Pi_{\pi}(s, s') z(s')^2 \leq \sum_s d_{\pi}(s) \Pi_{\pi}(s, s') z(s')^2 = \|z\|_{D_{\pi}}^2,
\]
where the first inequality comes from Jensen’s inequality, whose equality holds iff all components of \( z \) are the same scalar (referred to as \( z_c \neq 0 \)). When that happens, we have \( \|\Pi_{\pi} z\|_{D_{\pi}}^2 = z_c^2 \sum_s d_{\pi}(s) \gamma(s)^2 \). Note there exists at least one \( s \) such that \( \gamma(s) < 1 \), otherwise \( \Pi_{\pi} - I \) is singular, violating Assumption 1. So \( \|\Pi_{\pi} z\|_{D_{\pi}}^2 < z_c^2 = \|z\|_{D_{\pi}}^2 \). To conclude, for any \( z \), we always have \( \|\Pi_{\pi} z\|_{D_{\pi}} < \|z\|_{D_{\pi}} \), yielding
\[
z^\top D_{\pi} \Pi_{\pi} z \leq \|z\|_{D_{\pi}}^2 \|D_{\pi} \Pi_{\pi} z\| = \|z\|_{D_{\pi}} \|\Pi_{\pi} z\|_{D_{\pi}} = z^\top D_{\pi} \Pi_{\pi} z,
\]
which completes the proof.

Assumption 2(d) is straightforward as \( \mathcal{Y} \) is finite. Assumption 2(e) is trivial in our setting as we do not have eligibility trace and can be obtained from standard arguments about the mixing time of MDP (e.g., Theorem 4.9 in Levin and Peres (2017)).

The extension from deterministic rewards to stochastic rewards is standard (e.g., see Section 2.2 in Borkar (2009)) thus omitted.

**B.3 Proof of Proposition 2**

**Proof.** Assumption 1 implies \( \rho(P_{\pi} \top \Gamma) < 1 \). Then Lemma 1 implies that there exists a \( w \) in \( \mathbb{R}^{|s|} \) such that \( k_0 \geq \|P_{\pi} \top \Gamma\|_\infty < 1 \). Let \( \epsilon_2 \) be the Cramér distance in \( P(\mathbb{R}) \) (see Definition 3 in Rowland et al.
As we consider off-policy setting, the stationary distribution of \( \pi \) implies that \( \bar{\eta} \) is indeed a valid metric. Eq (6) then implies that \( \bar{T} \) is a \( \sqrt{k_0} \)-contraction in \( \bar{d} \). Standard fixed point theories then imply that \( \bar{T} \) has a unique fixed point, which we refer to as \( \eta_\pi \).

As \( \mu_0 = d_\pi \), we have \( p(s, r|s) = \Pr(S_{t-1} = \bar{s}, R_t = r|S_t = s) \) holds for all \( t \). Then Eq (2) implies that \( \eta_t = \bar{T}\eta_{t-1} \), from which \( \lim_{t \to \infty} d(\eta_t, \eta_\pi) = 0 \) follows directly.

**B.4 Proof of Proposition 3**

**Proof.** The proof is the same as the proof of Theorem 1 except that we define

\[
A(y) = \tau(s)p(s, a)(\gamma(s)x(s')x(s')^T - x(s)x(s')^T),
\]

\[
b(y) = \tau(s)p(s, a)r(s, a)x(s').
\]

As we consider off-policy setting, the stationary distribution of \( \{Y_t\} \) is then \( p_Y(y) = d_\mu(s)\mu(a|s)p(s'|s, a) \). It is easy to verify that we still have \( \bar{A} = \mathbb{E}_{y \sim p_Y}[A(y)] = X^\top(\pi^\top \Gamma I)D_xX \) and \( \bar{b} = \mathbb{E}_{y \sim p_Y}[b(y)] = \bar{P}_\pi^\top \bar{D}_x\bar{r} \). The rest is thus the same.

**C.** **Experiment Details**

**C.1 Anomaly Detection**

The TD3 agent used for generating \( \mu_\pi \) is the same as Fujimoto et al. (2018), which is trained for \( 2 \times 10^4 \) steps in Reacher to achieve an average episodic return of \( -4 \). We use two hidden-layer neural networks with ReLU activation function. Each hidden layer consists of 64 units. The output layer has 20 units, representing 20 quantiles. We use an Adam optimizer with an initial learning rate \( 5 \times 10^{-3} \). The size of the experience replay buffer is \( 10^6 \) and the mini-batch size is 128. We update the target network every 100 steps. We conduct our experiments on an Nvidia DGX-1 with PyTorch, though no GPU is used.
C.2 Representation Learning

As our IMPALA+ReverseGVF agent is simply replacing the canonical TD loss in the IMPALA+GVF agent with the Reverse TD loss, we refer the reader to [Veeriah et al. (2019)] for all the implementation details.