A New Q(λ)

Abstract

Q-learning, the most popular of reinforcement learning algorithms, has always included an extension to eligibility traces to enable more rapid learning and improved asymptotic performance on non-Markov problems. The λ parameter smoothly shifts on-policy algorithms such as TD(λ) and Sarsa(λ) from a pure bootstrapping form ($\lambda = 0$) to a pure Monte Carlo form ($\lambda = 1$). In off-policy algorithms, including Watkins's Q(λ), Peng's Q(λ), and the recent GQ(λ), the λ parameter is intended to play the same role, but does not; on every exploratory action these algorithms bootstrap absolutely regardless of the value of λ, and as a result they never approximate pure Monte Carlo learning. It may seem that this is inevitable for any online off-policy algorithm; if updates are made on each step on which the target policy is followed, then how could just the right updates be 'unmade' upon deviation from the target policy? In this paper, we introduce a new version of Q(λ) that does exactly that, without significantly increased algorithmic complexity. En route to our new Q(λ), we introduce a new derivation technique based on the forward-view/backward view analysis familiar from TD(λ) but extended to apply at every time step rather than only at the end of episodes. We apply this technique to derive a new off-policy TD(λ) and then our new Q(λ).

1. Off-policy eligibility traces

Eligibility traces (Sutton 1988, Singh & Sutton 1996) are the mechanism by which temporal-difference (TD) algorithms such as Q-learning (Watkins 1989) and Sarsa (Rummery 1995) escape the tyranny of the time step. In their simplest, one-step forms, these algorithms pass credit for an error back to only the single state preceding the error. If the time step is short, then the backward propagation of accurate values can be quite slow. With traces, credit is passed back to multiple preceding states, and learning is often significantly faster.

In on-policy TD algorithms with traces, such as TD(λ) and Sarsa(λ), the assignment of credit to previous states fades for more distantly preceding states according to the product of the bootstrapping parameter $\lambda \in [0, 1]$. If $\lambda = 1$, then the traces fade maximally slowly, no bootstrapping is done, and the resulting algorithm is called a *Monte Carlo* algorithm because in the limit it behaves almost exactly as if learning was to the complete observed return. If $\lambda = 0$, then the one-step form of the algorithm is recovered.

Many off-policy TD algorithms also use traces, but less successfully. Watkins's (1989) Q(λ), for example, assigns credit to preceding state–action pairs, fading with temporal distance, but only up until the last non-greedy action. If exploratory actions are common, then the traces rarely last very long. In part to redress this, Peng's (1993) Q(λ) never cuts the traces, but fails to converge to the optimal value function and still involves bootstrapping irrespective of λ. A more important problem with Watkins's Q(λ) is that it bootstraps (updates its estimates from other estimates) whenever an exploratory action is taken. GQ(λ) (Maei & Sutton 2010) and GTD(λ) (Maei 2011) extend Q(λ) to function approximation in a sound way and handle off-policy learning more generally, but have essentially the same weakness: when the action taken deviates from the target policy they all cut off their traces and bootstrap. One consequence is that none of these algorithms can approximate Monte Carlo algorithms. More generally, the degree of bootstrapping cannot be set by the user (via λ) independently of the actions taken.

It might seem impossible for an online off-policy TD algorithm to do anything other than bootstrap when the action taken deviates from the target policy. By the time of deviation many online updates have already been made, but the deviation means the subsequent rewards cannot be counted as due to the target policy. The right thing to do, if $\lambda = 1$, is to somehow undo the earlier updates, but this has always seemed impossible to do online without vastly increasing the computational complexity of the algorithm. In this paper we show that, surprisingly, it can be done with very little additional computation. In the case of linear function approximation, one additional vector is needed to hold *provisional weights*, the portion of the main weights that may

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

need to be removed upon deviation, or otherwise adjusted based on the degree of match to the target policy.

A key part of previous analyses of eligibility traces has been the notion of a *forward view* specifying the ideal update in terms of future events (Sutton & Barto 1998, Chapter 7). It can then sometimes be shown that a causal algorithm (a *backward view*) is mathematically equivalent in its overall effect to the forward view. Our first contribution is to introduce a new notion of "interim" forward view for general off-policy learning and for arbitrary λ and γ that are not just parameters but general functions of state. Our second contribution is a powerful technique for analytically deriving new TD algorithms that are equivalent to interim forward views. We first illustrate the technique by using it to show that conventional TD(λ) is equivalent to the on-policy special case of the interim forward view (itself a new result), then use it to derive and prove the equivalence of a new off-policy TD(λ) algorithm. Finally, we introduce an interim forward view for action values and use it to derive and prove equivalence of our new Q(λ). Like the original equivalences, ours are exact for offline updating and approximate for online updating. Our algorithms are all described in terms of linear function approximation but, like the original Q-learning, they are guaranteed convergent only for the tabular case; the extension to gradient-TD methods seems straightforward but is left to future work.

2. A general off-policy forward view

We consider a continuing setting in which an agent and environment interact at each of a series of time steps, $t = 0, 1, 2, \ldots$. At each step t, the environment is in state S_t and generates a feature vector $\boldsymbol{\phi}(S_t) \in \mathbb{R}^d$. The agent then chooses an action A_t from a distribution defined by its fixed *behavior policy* $b(\cdot|S_t)$ (the only influence of S_t on b is typically assumed to be via $\boldsymbol{\phi}(S_t)$, but we do not enforce this). The environment then emits a reward $R_{t+1} \in \mathbb{R}$ and transitions to a new state S_{t+1}, and the process continues. The next state and reward are assumed to be chosen from a joint distribution that depends only on the preceding state and action (the Markov assumption). We first consider the problem of finding a weight vector $\boldsymbol{\theta} \in \mathbb{R}^d$ such that

$$\boldsymbol{\theta}^\top \boldsymbol{\phi}(s) \approx \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} R_{t+1} \prod_{k=1}^{t} \gamma(S_k) \middle| S_0 = s \right] \qquad (1)$$

in some sense (for some norm), where the expectation is conditional on actions being selected according to an alternative policy π, called the *target policy*, and $\gamma(\cdot)$ is an almost arbitrary function from the state space to $[0, 1]$. We will refer to $\gamma(\cdot)$ as the *termination function* because when it falls to zero it terminates the quantity whose expectation is being estimated. The only restriction we place on the termination function (and on the environment) is that $\prod_{k=0}^{\infty} \gamma(S_{t+k}) = 0$ w.p.1, $\forall t$.

If the two policies are the same, $\pi = b$, then the setting is called on-policy. If $\pi \neq b$, then we have the off-policy case, which is harder. In general, π and b may also vary over time, e.g. in the case of control algorithms which rely on soft exploration. Normally it is assumed that the behavior policy b overlaps the target policy π in the sense that $\pi(a|s) > 0 \implies b(a|s) > 0$. At each time t the degree of deviation of the target policy from the behavior policy is captured by the importance-sampling ratio (Precup et al. 2000, 2001):

$$\rho_t = \frac{\pi(A_t|S_t)}{b(A_t|S_t)}. \qquad (2)$$

It is awkward to seek the approximation (1) from data in the off-policy case because the expectation is under π whereas the data is due to b. Some actions chosen by b will match π, but not an infinite number in succession. This is where it is useful to interpret $\gamma(\cdot)$ as termination rather than as discounting. That is, we consider any trajectory, for example, $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$ as having terminated partly after one step, at S_1, with degree of termination $1-\gamma(S_1)$, yielding a return of R_1, and partly after two steps, at S_2, with degree of termination $\gamma(S_1)(1-\gamma(S_2))$, yielding a return of $R_1 + R_2$. The third partial termination would have a degree of termination of $\gamma_1\gamma_2(1-\gamma_3)$ (where here we have switched to the shorthand $\gamma_t = \gamma(S_t)$) and yield a return of $R_1 + R_2 + R_3$. Notice how in this partial termination view we end up with undiscounted, or *flat*, returns with no factors involving $\gamma(\cdot)$ in them. More importantly, we get short returns that have actually terminated, to various degrees, after each finite number of steps, rather that having to wait for the infinite number of steps to get a return as suggested by the discounting perspective.

Now consider the off-policy aspect and the role of the importance-sampling ratios. The first return, consisting of just R_1, need to be weighted not just by its degree of termination, $1-\gamma_1$, but also by the importance sampling ratio ρ_0. This ratio will emphasize or de-emphasize this return depending on whether the action A_0 was more or less common under π than it is under b. If it was an action that would rarely be done under the target policy π and yet is in fact common (under the behavior policy b) then this instance of it will be discounted proportionally to make up for its prevalence. If it would be common under the target policy but is rare (under the behavior policy) then its weighting should be pumped up proportionally to balance its infrequency. The overall weight on this return should then be the product of the degree of termination and the importance sampling ratio, or $\rho_0(1-\gamma_1)$. The weighting of the second return, $R_1 + R_2$, will depend on its degree of termination, $\gamma_1(1-\gamma_2)$ times the product of two importance sampling ratios, $\rho_0\rho_1$. The first takes into ac-

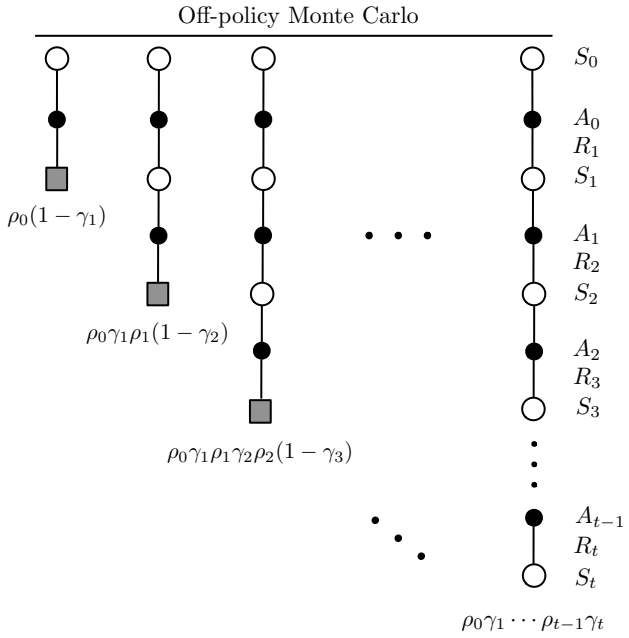

Figure 1. The complex backup for an off-policy Monte Carlo algorithm, illustrating weighting by degree of termination and degree of match of target and behavior policies. Also illustrated is the notion of an interim forward view, going only up to horizon t, at which time full bootstrapping is done. The component backups all use flat returns without discounting.

count the match of the target and behavior policies for A_0, and the second takes into account the match of the policies for A_1. The overall weighting on the second term is $\rho_0\gamma_1\rho_1(1-\gamma_2)$. Continuing similarly, the weighting on the third flat return, $R_1 + R_2 + R_3$ is $\rho_0\gamma_1\rho_1\gamma_2\rho_2(1-\gamma_3)$.

These weights and the backup diagrams for these three returns are shown in the first three columns of Figure 1. This is the standard way of diagramming a "complex" backup—a backup composed of of multiple sub-backups, one per column, each done in parallel with the weighting written below them (Sutton & Barto 1998, Chapter 7). An important difference, however, is that here the composite backups all use flat backups, without discounting or embedded termination, whereas Sutton and Barto always include a scalar γ implicit in their backup diagrams. The last column of the diagram is another small innovation. This backup is an *interim forward view*, meaning that it looks ahead not infinitely, but only out to some finite horizon time t. The horizon could be chosen arbitrarily far out in the future, so in this limiting sense interim forward views are a generalization of conventional forward views. Normally, however, we will be choosing the horizon far short of infinity. Normally we will chose t to be something like the current limit of available data—the current time. Interim forward views are used to talk about the updates that could have been done in the past using all the data up to the current time. Note that this last backup is to a state rather than termination. According to the convention of backup diagrams, this means that bootstrapping using the current approximation is done at time t.

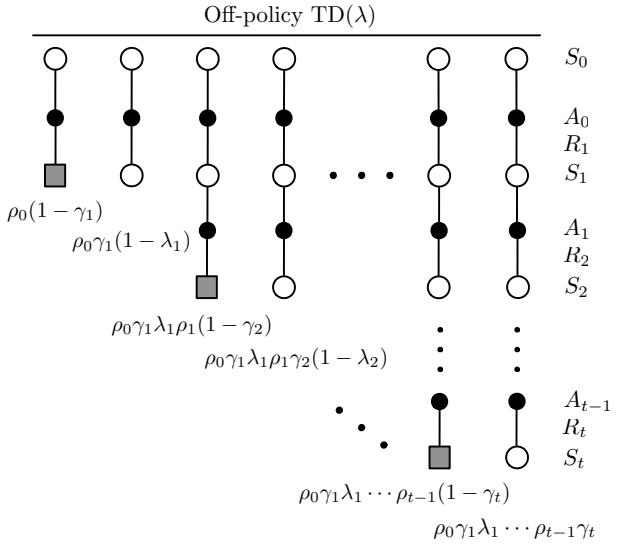

Figure 2. The complex backup for the new off-policy TD(λ) shows the weighting on each component n-step backup, using flat returns.

Next we introduce bootstrapping into our general forward view. We follow Sutton and Barto (1998, Section 7.10) and generalize the conventional scalar λ parameter to an arbitrary bootstrapping function $\lambda(\cdot)$ from the state space to $[0, 1]$. The bootstrapping factor at time t, $\lambda_t = \lambda(S_t)$, represents the degree of bootstrapping upon arrival in S_t, just as the termination factor $\gamma_t = \gamma(S_t)$ represents the degree of termination upon arriving there. The new backup diagram is shown in Figure 2. Each terminating sub-backup of the Monte Carlo backup in Figure 1 has been split into a terminating sub-backup and a bootstrapping sub-backup. Each sub-backup is a way in which the return can finish and no longer be subject to reweightings based on subsequent importance sampling ratios. The overall backup is best read from left to right. In the first two columns, the return can terminate in S_1 with a weighting of $\rho_0(1-\gamma_1)$, yielding a return of just R_1 or, to the extent that that does not occur, it can bootstrap in S_1 yielding a return of $R_1 + \boldsymbol{\theta}^\top\boldsymbol{\phi}(S_1)$. Bootstrapping can only be done to the extent that there was no prior deviation, ρ_0, or termination, γ_1, and only to the extent that λ_1 is less than 1; the overall weighting is thus $\rho\gamma_1(1-\lambda_1)$. To the extent that neither of these occur we go on similarly to the two-step sub-backups, which yield returns of $R_1 + R_2$ or $R_1 + R_2 + \boldsymbol{\theta}^\top\boldsymbol{\phi}(S_2)$ respectively if they apply, or we go on to the three-step sub-backups, and so on until we reach the last bootstrapping sub-backup at the horizon, at which bootstrapping is complete as if $\lambda_t = 1$, yielding a return of $R_1 + \cdots + R_t + \boldsymbol{\theta}^\top\boldsymbol{\phi}(S_t)$ with weighting $\rho_0\gamma_1\lambda_1\cdots\lambda_{t-1}\rho_{t-1}\gamma_t$.

There is one final important issue before we can reduce these ideas and backup diagrams to equations. The preceding discussion has suggested that there is a total amount of weight that must be split up among the component backups of the complex backup. This is not totally incorrect. The expectation of the total weight along a trajectory is always one by construction, but the total weight for sample trajectory will often be significantly larger or smaller than one. This is entirely caused by the importance sampling ratios. Previous complex backups, including off-policy ones like Q-learning, have not had to deal with this issue. We deal with it here by switching from viewing the backup diagram as a way of generating a *return* to a way of generating an *error*. We will compute the weighting and then use it to weight the complete error rather than just the return portion of the error. This change does not affect the expected update of the algorithm, but we believe that it significantly reduces the average variance (Precup et al. 2001).

As an example of the issue, consider a case where the first action substantially deviates from the target policy, say $\rho_0 = 0.01$, and then terminates, $\gamma_1 = 0$. This yields a return of just $0.01 \cdot R_1$. Conversely, suppose the first action strongly matches the target policy with $\rho_0 = 10$ and then termination occurs. This yields a target of $10 \cdot R_1$. If we view the weighting as on these returns, then the errors $0.01 \cdot R_1 - \boldsymbol{\theta}^\top \boldsymbol{\phi}(S_0)$ and $10 \cdot R_1 - \boldsymbol{\theta}^\top \boldsymbol{\phi}(S_0)$ we will move the estimate $\boldsymbol{\theta}^\top \boldsymbol{\phi}(S_0)$ in one case toward nearly zero and in the other case towards a potentially large number, even if R_1 is the same in the two cases. If instead we view the weighing as on the errors, then both errors will be $R_1 - \boldsymbol{\theta}^\top \boldsymbol{\phi}(S_0)$, only one will be weighted by 0.01 and the other by 10. The updates will be different sizes, but not gratuitously in different directions. This specific example is contrived of course, with termination after just one step, but widely differing weightings are in fact not at all atypical here due to the products of multiple importance-sampling ratios.

In our complex backup (Figure 2) there are two kinds of sub-backups. The first, ending with a terminal state, corresponds to an n-step error without bootstrapping:

$$\epsilon_t^{(n)} = R_{t+1} + R_{t+2} + \cdots + R_{t+n} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(S_t). \tag{3}$$

The second sub-backup of each pair, ending with a non-terminal state, corresponds to an n-step flat TD error, that is, an n-step error with bootstrapping but no discounting or termination:

$$\bar{\delta}_t^{(n)} = R_{t+1} + \cdots + R_{t+n} + \boldsymbol{\theta}^\top \boldsymbol{\phi}(S_{t+n}) - \boldsymbol{\theta}^\top \boldsymbol{\phi}(S_t). \tag{4}$$

Using these, the overall error corresponding to the forward view in Figure 2, which we denote δ_t^ρ, can be written compactly as a nested sum of pairs of terms, each corresponding to a sub-backup of the figure:

$$\begin{aligned}
\delta_{0,t}^\rho &= \rho_0 \Big[(1-\gamma_1)\epsilon_0^{(1)} + \gamma_1(1-\lambda_1)\bar{\delta}_0^{(1)} \\
&\quad + \gamma_1\lambda_1\rho_1 \Big[(1-\gamma_2)\epsilon_0^{(2)} + \gamma_2(1-\lambda_2)\bar{\delta}_0^{(2)} \\
&\quad + \gamma_2\lambda_2\rho_2 \Big[(1-\gamma_3)\epsilon_0^{(3)} + \gamma_3(1-\lambda_3)\bar{\delta}_0^{(3)} \\
&\quad + \cdots \gamma_{t-1}\lambda_{t-1}\rho_{t-1} \big[(1-\gamma_t)\epsilon_0^{(t)} + \gamma_t \bar{\delta}_0^{(t)} \big] \Big] \Big] \Big] \\
&= \rho_0 \sum_{n=1}^{t} D_0^{(n-1)} \Big((1-\gamma_n)\epsilon_0^{(n)} + \gamma_n(1-\lambda_n^t)\bar{\delta}_0^{(n)} \Big).
\end{aligned}$$

where λ_n^t is syntactic sugar for λ_n except for $n = t$, in which case it is defined to be 0. We denote $D_0^{(n-1)} = \prod_{i=1}^{n-1} \gamma_i \lambda_i \rho_i$. Note that in general $D_t^{(0)} = 1$ and

$$D_t^{(n)} = \gamma_{t+1}\lambda_{t+1}\rho_{t+1} D_{t+1}^{(n-1)}. \tag{5}$$

3. Analysis of the forward view

In this section we analyze and present several formal results for the general forward view. Some of these will be used to simplify computations later and some are of independent interest. First, we define

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}(S_t) \tag{6}$$

$$\delta_t = R_{t+1} + \gamma_{t+1} \boldsymbol{\theta}^\top \boldsymbol{\phi}_{t+1} - \boldsymbol{\theta}^\top \boldsymbol{\phi}_t \tag{7}$$

and note the following obvious but useful identities:

$$\delta_t = \epsilon_t^{(1)} + \gamma_{t+1} \boldsymbol{\theta}^\top \boldsymbol{\phi}_{t+1} \tag{8}$$

$$\bar{\delta}_t^{(n)} = \epsilon_t^{(n)} + \boldsymbol{\theta}^\top \boldsymbol{\phi}_{t+n} \tag{9}$$

$$\bar{\delta}_t^{(n)} = \bar{\delta}_{t+1}^{(n-1)} + \bar{\delta}_t^{(1)} = \bar{\delta}_t^{(n-1)} + \bar{\delta}_{t+n-1}^{(1)} \tag{10}$$

$$\epsilon_t^{(n)} = \epsilon_{t+1}^{(n-1)} + \bar{\delta}_t^{(1)} = \bar{\delta}_t^{(n-1)} + \epsilon_{t+n-1}^{(1)}. \tag{11}$$

Using these, we can rewrite $\delta_{k,t}^\rho$ (for any $0 \le k < t$) in a useful recursive form:

$$\delta_{k,t}^\rho = \rho_k \sum_{n=1}^{t-k} D_k^{(n-1)} \Big(\epsilon_k^{(n)} + \gamma_{k+n} \boldsymbol{\theta}^\top \boldsymbol{\phi}_{k+n} - \gamma_{k+n} \lambda_{k+n}^t \bar{\delta}_k^{(n)} \Big) \tag{12}$$

$$\begin{aligned}
&= \rho_k \delta_k - \rho_k \gamma_{k+1} \lambda_{k+1}^t \bar{\delta}_k^{(1)} \\
&\quad + \rho_k \sum_{n=2}^{t-k} \gamma_{k+1} \lambda_{k+1}^t \rho_{k+1} D_{k+1}^{(n-2)} \Big(\gamma_{k+n} \boldsymbol{\theta}^\top \boldsymbol{\phi}_{k+n} \\
&\quad + \epsilon_{k+1}^{(n-1)} + \bar{\delta}_k^{(1)} - \gamma_{k+n} \lambda_{k+n}^t (\bar{\delta}_{k+1}^{(n-1)} + \bar{\delta}_k^{(1)}) \Big) \\
&= \rho_k \Big(\delta_k + \gamma_{k+1} \lambda_{k+1}^t \delta_{k+1,t}^\rho + \gamma_{k+1} \lambda_{k+1}^t (C_{k+1,t} - 1) \bar{\delta}_k^{(1)} \Big)
\end{aligned} \tag{13}$$

where:

$$C_{k,t} = \sum_{n=1}^{t-k} D_k^{(n-1)}(1-\gamma_{k+n}\lambda_{k+n}^t) \quad (14)$$

$$= \sum_{n=1}^{t-k}(1-\gamma_{k+n}\lambda_{k+n}^t)\prod_{i=1}^{n-1}\gamma_{k+i}\lambda_{k+i}^t\rho_{k+i} \quad (15)$$

We now show that the on-policy forward-view of TD(λ) with general γ and λ is a special case of the off-policy TD(λ) forward view.

Lemma 1. *If $\forall l$, $\rho_l = 1$, then: $C_{k,t} = 1, \forall k < t, t$*

Proof. Because $\rho_l = 1, \forall l$, we have: $C_{k,t} = \sum_{n=1}^{t-k}(1-\gamma_{k+n}\lambda_{k+n}^t)\prod_{i=1}^{n-1}\gamma_{k+i}\lambda_{k+i}^t$. It is easy to prove by induction that:

$$C_{k,t} = 1 - \prod_{n=1}^{t-k}\gamma_{k+n}\lambda_{k+n}^t$$

Because $\lambda_t^t = 0$ by definition, $C_{k,t} = 1$. □

We now show that this algorithm is the same as usual TD(λ) in the on-policy case. Let $\delta_k^\rho = \lim_{t\to\infty}\delta_{k,t}^\rho$ Recall that the on-policy forward-view update of TD(λ) can be written as

$$\alpha\delta_k^\lambda\boldsymbol{\phi}_k = \alpha(G_k^\lambda - \boldsymbol{\theta}^\top\boldsymbol{\phi}_k)\boldsymbol{\phi}_k, \quad (16)$$

where the λ-return, G_k^λ is defined recursive as (Maei, 2011):

$$G_k^\lambda = R_{k+1} + \gamma_{k+1}\left((1-\lambda_{k+1})\boldsymbol{\theta}^\top\boldsymbol{\phi}_{k+1} + \lambda_{k+1}G_{k+1}^\lambda\right).$$

so the on-policy forward-view error can be written as

$$\delta_k^\lambda = G_k^\lambda - \boldsymbol{\theta}^\top\boldsymbol{\phi}_k\delta_k + \gamma_{k+1}\lambda_{k+1}\left(G_{k+1}^\lambda - \boldsymbol{\theta}^\top\boldsymbol{\phi}_{k+1}\right)$$
$$= \delta_k + \gamma_{k+1}\lambda_{k+1}\delta_{k+1}^\lambda. \quad (17)$$

Theorem 1. *If $\rho_l = 1\ \forall l$, then the forward-view (12) is equivalent to the on-policy forward-view update of TD(λ) defined in (16).*

Proof. If $\rho_l = 1\ \forall l$, from (13) and Lemma 1 taking the limit as $t \to \infty$,we have:

$$\delta_k^\rho = \delta_k + \gamma_{k+1}\lambda_{k+1}\delta_{k+1}^\rho.$$

Hence, the two forward-view updates coincide. □

We now briefly examine the correctness of the off-policy algorithm.

Lemma 2. $\mathbb{E}_b[C_{k,t}] = 1$.

Proof. Follows immediately from the definitions of ρ and $C_{k,t}$ and Lemma 1. □

Theorem 2. *Let b and π be the behavior and the target policies, respectively. Then, in expectation, the off-policy forward-view (13) under the behavior policy achieves the on-policy forward-view (16) under the target policy, that is,*

$$\mathbb{E}_b\left[\delta_{k,t}^\rho\boldsymbol{\phi}_k\right] = \mathbb{E}_\pi\left[\delta_{k,t}^\lambda\boldsymbol{\phi}_k\right].$$

Proof. Follows immediately from (13), Lemma 2 and Eqs.(17). □

4. Interim equivalence of forward and backward views

Based on the forward view defined in Sec. 2 and its general form (13), the ideal forward-view update at time k given only the portion of the trajectory up through S_t, $\Delta\boldsymbol{\theta}_{k,t}^F$, is given by:

$$\Delta\boldsymbol{\theta}_{k,t}^F = \alpha\delta_{k,t}^\rho\boldsymbol{\phi}_k \quad (18)$$

We now introduce an interim equivalence technique for deriving an equivalent backward-view algorithm from a forward-view algorithm. If the weight vector is held at $\boldsymbol{\theta}$, it follows then that the ideal (off-line) forward-view weight vector, given only the data up through S_t, is

$$\boldsymbol{\theta}_t^* = \boldsymbol{\theta} + \sum_{k=0}^{t-1}\Delta\boldsymbol{\theta}_{k,t}^F,$$

and that the corresponding ideal update, from t to $t+1$, is

$$\Delta\boldsymbol{\theta}_t^* = \boldsymbol{\theta}_{t+1}^* - \boldsymbol{\theta}_t^* = \sum_{k=0}^{t}\Delta\boldsymbol{\theta}_{k,t+1}^F - \sum_{k=0}^{t-1}\Delta\boldsymbol{\theta}_{k,t}^F.$$

Hence, one should simply work from the equation above to derive the backward-view update:

$$\Delta\boldsymbol{\theta}_t^B = \sum_{k=0}^{t}\Delta\boldsymbol{\theta}_{k,t+1}^F - \sum_{k=0}^{t-1}\Delta\boldsymbol{\theta}_{k,t}^F. \quad (19)$$

We now prove that the backward view obtained this way is equivalent to the forward view. This will be the cornerstone of all eligibility trace algorithms presented in the rest of the paper.

Theorem 3. *The total update generated by any backward view derived using (19) is equivalent to the total forward update up to time t:*

$$\forall t, \sum_{k=0}^{t-1}\Delta\boldsymbol{\theta}_k^B = \sum_{k=0}^{t-1}\Delta\boldsymbol{\theta}_{k,t}^F.$$

Proof: $\sum_{k=0}^{t-1}\Delta\boldsymbol{\theta}_k^B = \sum_{k=0}^{t-1}(\boldsymbol{\theta}_{k+1}^* - \boldsymbol{\theta}_k^*) = \boldsymbol{\theta}_t^* - \boldsymbol{\theta} = \sum_{k=0}^{t-1}\Delta\boldsymbol{\theta}_k^F$. □

We now illustrate how this interim equivalence technique can be applied to derive a backward view for the on-policy TD(λ) with general γ and λ. The forward-view of on-policy TD(λ) with general γ and λ can be defined as

$$\Delta\boldsymbol{\theta}^F_{k,t} = \alpha\delta^\lambda_{k,t}\boldsymbol{\phi}_k. \tag{20}$$

The truncated on-policy λ-TD error with general γ and λ, $\delta^\lambda_{k,t}$, is obtained by setting all $\rho = 1$ in (13), and taking into account Lemma 1, as:,

$$\delta^\lambda_{k,t} = \delta_k + \gamma_{k+1}\lambda^t_{k+1}\delta^\lambda_{k+1,t}.$$

To derive the backward view for the on-policy TD(λ) with general γ and λ, we can write:

$$\begin{aligned}
&\frac{1}{\alpha}\left(\sum_{k=0}^{t}\Delta\boldsymbol{\theta}^F_{k,t+1} - \sum_{k=0}^{t-1}\Delta\boldsymbol{\theta}^F_{k,t}\right) = \sum_{k=0}^{t}\delta^\lambda_{k,t+1}\boldsymbol{\phi}_k - \sum_{k=0}^{t-1}\delta^\lambda_{k,t}\boldsymbol{\phi}_k \\
&= \delta_t\boldsymbol{\phi}_t + \sum_{k=0}^{t-1}\left(\delta^\lambda_{k,t+1} - \delta^\lambda_{k,t}\right)\boldsymbol{\phi}_k \\
&= \delta_t\boldsymbol{\phi}_t + \sum_{k=0}^{t-1}\gamma_{k+1}\lambda^t_{k+1}\left(\delta^\lambda_{k+1,t+1} - \delta^\lambda_{k+1,t}\right)\boldsymbol{\phi}_k \\
&= \delta_t\boldsymbol{\phi}_t + \sum_{k=0}^{t-1}\left(\prod_{i=1}^{t-k-1}\gamma_{k+i}\lambda^t_{k+i}\right)\left(\delta^\lambda_{t-1,t+1} - \delta^\lambda_{t-1,t}\right)\boldsymbol{\phi}_k \\
&= \delta_t\boldsymbol{\phi}_t + \sum_{k=0}^{t-1}\left(\prod_{i=1}^{t-k-1}\gamma_{k+i}\lambda^t_{k+i}\right)\gamma_k\lambda^t_k\delta_k\boldsymbol{\phi}_k \\
&= \delta_t\boldsymbol{e}_t,
\end{aligned}$$

where we define $\boldsymbol{e}_t$ (the eligibility trace vector) as:

$$\begin{aligned}
\boldsymbol{e}_t &= \boldsymbol{\phi}_t + \sum_{k=0}^{t-1}\left(\prod_{i=1}^{t-k}\gamma_{k+i}\lambda^t_{k+i}\right)\boldsymbol{\phi}_k \\
&= \boldsymbol{\phi}_t + \gamma_t\lambda_t\left(\boldsymbol{\phi}_{t-1} + \sum_{k=0}^{t-2}\left(\prod_{i=1}^{t-k-1}\gamma_{k+i}\lambda^t_{k+i}\right)\boldsymbol{\phi}_k\right) \\
&= \boldsymbol{\phi}_t + \gamma_t\lambda_t\boldsymbol{e}_{t-1}.
\end{aligned}$$

Therefore,

$$\Delta\boldsymbol{\theta}^B_t = \sum_{k=0}^{t}\Delta\boldsymbol{\theta}^F_{k,t+1} - \sum_{k=0}^{t-1}\Delta\boldsymbol{\theta}^F_{k,t} = \alpha\delta_t\boldsymbol{e}_t, \tag{21}$$

which is the on-policy backward view update of TD(λ) with general γ and λ. Note that, for constant γ and λ, this update becomes the conventional on-policy backward view update of TD(λ).

5. Derivation of a New TD(λ)

We now use the same interim equivalence technique to derive our new off-policy TD(λ) backward-view algorithm, starting from the forward view (12):

$$\begin{aligned}
\delta^\rho_{k,t} &= \rho_k D_k^{(t-k-1)}\left((1-\gamma_t)\epsilon_k^{(t-k)} + \gamma_t\bar{\delta}_k^{(t-k)}\right) \\
&+ \sum_{n=1}^{t-k-1}\rho_k D_k^{(n-1)}\left((1-\gamma_{k+n})\epsilon_k^{(n)} + \gamma_{k+n}(1-\lambda_{k+n})\bar{\delta}_k^{(n)}\right).
\end{aligned}$$

Using (8), (10) and (11), we can write it recursively as

$$\begin{aligned}
\delta^\rho_{k,t} &= \rho_k D_k^{(t-k-1)}\delta_{t-1} + \rho_k\gamma_{t-1}\lambda_{t-1}\rho_{t-1}D_k^{(t-k-2)}\bar{\delta}_k^{(t-k-1)} \\
&\quad - \rho_k\gamma_{t-1}\lambda_{t-1}D_k^{(t-k-2)}\bar{\delta}_k^{(t-k-1)} \\
&\quad + \rho_k D_k^{(t-k-2)}\left((1-\gamma_{t-1})\epsilon_k^{t-k-1} + \gamma_{t-1}\bar{\delta}_k^{(t-k-1)}\right) \\
&\quad + \sum_{n=1}^{t-k-2}\rho_k D_k^{(n-1)}\left((1-\gamma_{k+n})\epsilon_k^{n} + \gamma_{k+n}(1-\lambda_{k+n})\bar{\delta}_k^{(n)}\right) \\
&= \delta^\rho_{k,t-1} + \rho_k D_k^{t-k-1}\delta_{t-1} \\
&\quad + (\rho_{t-1}-1)\gamma_{t-1}\lambda_{t-1}\rho_k D_k^{t-k-2}\bar{\delta}_k^{(t-k-1)}.
\end{aligned}$$

The forward-view update at k, given only the data up through S_t, is

$$\Delta\boldsymbol{\theta}^F_{k,t} = \alpha\delta^\rho_{k,t}\boldsymbol{\phi}_k. \tag{22}$$

Then to derive the backward view, we can use the same technique that we developed in the previous section and write:

$$\begin{aligned}
&\frac{1}{\alpha}\left(\sum_{k=0}^{t}\Delta\boldsymbol{\theta}^F_{k,t+1} - \sum_{k=0}^{t-1}\Delta\boldsymbol{\theta}^F_{k,t}\right) \\
&= \delta^\rho_{t,t+1}\boldsymbol{\phi}_t + \sum_{k=0}^{t-1}\left[\delta^\rho_{k,t+1} - \delta^\rho_{k,t}\right]\boldsymbol{\phi}_k \\
&= \rho_t\delta_t\boldsymbol{\phi}_t \\
&+ \sum_{k=0}^{t-1}\left[\rho_k D_k^{(t-k)}\delta_t + (\rho_t-1)\gamma_t\lambda_t\rho_k D_k^{(t-k-1)}\bar{\delta}_k^{(t-k)}\right]\boldsymbol{\phi}_k \\
&= \delta_t\boldsymbol{e}_t + (\rho_t-1)\boldsymbol{u}_t.
\end{aligned}$$

We define $\boldsymbol{e}_t$ as

$$\begin{aligned}
\boldsymbol{e}_t &= \rho_t\boldsymbol{\phi}_t + \sum_{k=0}^{t-1}\rho_k D_k^{(t-k)}\boldsymbol{\phi}_k \\
&= \rho_t\boldsymbol{\phi}_t + \gamma_t\lambda_t\rho_t\left(\rho_{t-1}\boldsymbol{\phi}_{t-1} + \sum_{k=0}^{t-2}\rho_k D_k^{(t-k-1)}\boldsymbol{\phi}_k\right) \\
&= \rho_t\left(\boldsymbol{\phi}_t + \gamma_t\lambda_t\boldsymbol{e}_{t-1}\right),
\end{aligned} \tag{23}$$

and $\boldsymbol{u}_t$ as

$$\begin{aligned}
\boldsymbol{u}_t &= \gamma_t\lambda_t\sum_{k=0}^{t-1}\rho_k D_k^{(t-k-1)}\bar{\delta}_k^{(t-k)}\boldsymbol{\phi}_k \\
&= \gamma_t\lambda_t\left(\rho_{t-1}\gamma_{t-1}\lambda_{t-1}\sum_{k=0}^{t-2}\rho_k D_k^{(t-k-2)}\bar{\delta}_k^{(t-k-1)}\boldsymbol{\phi}_k\right.
\end{aligned}$$

$$+\sum_{k=0}^{t-2}\rho_k D_k^{(t-k-1)}\bar{\delta}_{t-1}^{(1)}\boldsymbol{\phi}_k+\rho_{t-1}\bar{\delta}_{t-1}^{(1)}\boldsymbol{\phi}_{t-1}\Big)$$
$$= \gamma_t\lambda_t\left(\rho_{t-1}\boldsymbol{u}_{t-1}+\bar{\delta}_{t-1}^{(1)}\boldsymbol{e}_{t-1}\right).$$

Let us redefine $\boldsymbol{u}_t=\alpha\boldsymbol{u}_t$. Then we can write

$$\boldsymbol{u}_t=\gamma_t\lambda_t\left(\rho_{t-1}\boldsymbol{u}_{t-1}+\alpha\bar{\delta}_{t-1}^{(1)}\boldsymbol{e}_{t-1}\right). \tag{24}$$

Then the off-policy backward-view update of TD(λ) with general γ and λ can be written as

$$\Delta\boldsymbol{\theta}_t^B=\sum_{k=0}^{t}\Delta\boldsymbol{\theta}_{k,t+1}^F-\sum_{k=0}^{t-1}\Delta\boldsymbol{\theta}_{k,t}^F=\alpha\delta_t\boldsymbol{e}_t+(\rho_t-1)\boldsymbol{u}_t. \tag{25}$$

Note that, for $\rho_t=1\ \forall t$, the last term in the weight update (25) is eliminated, and this algorithm becomes the on-policy TD(λ) update with general γ and λ.

Theorem 4. *The backward view algorithm defined by (23), (24) and (25) is equivalent to the forward-view defined in (22).*

Proof. The proof follows from Theorem 3 and the above derivation. □

6. Derivation of a new Q(λ)

We now extend the off-policy forward-view for action value estimation. Here, we consider a given target policy, although the algorithm readily extends to the case of control, by considering the target policy to be changing with time, for example, greedy with respect to the current action value estimates. We assume that feature vectors are now defined over state action-pairs, rather than states. The learning problem is to find a parameter vector $\boldsymbol{\theta}$ such that:

$$\boldsymbol{\theta}^\top\boldsymbol{\phi}(s,a)\approx\mathbb{E}_\pi\left[G_t^\lambda\middle|S_t=s,A_t=a\right],$$

As before, for brevity we denote $\boldsymbol{\phi}_t=\boldsymbol{\phi}(S_t,A_t)$. The intuition for the forward view is given in Fig. 3. This is the same as Fig. 2 with two differences: trajectories start with state-action pairs, and at the end, bootstrapping is done using the value of the state (considering all possible actions that the target policy may follow from then on):

$$\sum_a\pi(a|S_{t+1})\boldsymbol{\theta}^\top\boldsymbol{\phi}(S_{t+1},a)=\boldsymbol{\theta}^\top\bar{\boldsymbol{\phi}}_{t+1}^\pi,$$

where we denote by $\bar{\boldsymbol{\phi}}_t^\pi=\sum_a\pi(a|S_t)\boldsymbol{\phi}(S_t,a)$ the *expected feature* under policy π.

Let $\epsilon_t^{(n)}$ be the n-step undiscounted, uncorrected error and we redefine δ_k the usual TD-error for action values. Hence, it follows that $\delta_k=\epsilon_k^{(1)}+\gamma_{k+1}\boldsymbol{\theta}^\top\bar{\boldsymbol{\phi}}_{k+1}^\pi$ and $\epsilon_k^{(t-k)}+\gamma_t\boldsymbol{\theta}^\top\bar{\boldsymbol{\phi}}_t^\pi=\delta_{t-1}+\epsilon_k^{(t-k-1)}+\boldsymbol{\theta}^\top\boldsymbol{\phi}_{t-1}$.

Then following the intuition in Sec. 2, we redefine the forward-view TD error of the Q-value function as below:

$$\begin{aligned}\delta_{k,t}^\rho=&D_k^{(t-k-1)}(\epsilon_k^{(t-k)}+\gamma_t\boldsymbol{\theta}^\top\bar{\boldsymbol{\phi}}_t^\pi)\\&+\sum_{n=1}^{t-k-1}D_k^{(n-1)}\Big[(1-\gamma_{k+n}\lambda_{k+n})\epsilon_k^{(n)}\\&+\gamma_{k+n}(1-\lambda_{k+n})\boldsymbol{\theta}^\top\bar{\boldsymbol{\phi}}_{k+n}^\pi\Big].\end{aligned}$$

We can write $\delta_{k,t}^\rho$ recursively as

$$\begin{aligned}\delta_{k,t}^\rho=&D_k^{(t-k-1)}\delta_{t-1}\\&+\gamma_{t-1}\lambda_{t-1}\rho_{t-1}D_k^{(t-k-2)}\left(\boldsymbol{\theta}^\top\boldsymbol{\phi}_{t-1}-\boldsymbol{\theta}^\top\bar{\boldsymbol{\phi}}_{t-1}^\pi\right)\\&+\gamma_{t-1}\lambda_{t-1}\rho_{t-1}D_k^{(t-k-2)}\left(\epsilon_k^{(t-k-1)}+\boldsymbol{\theta}^\top\bar{\boldsymbol{\phi}}_{t-1}^\pi\right)\\&-\gamma_{t-1}\lambda_{t-1}D_k^{(t-k-2)}\left(\epsilon_k^{(t-k-1)}+\boldsymbol{\theta}^\top\bar{\boldsymbol{\phi}}_{t-1}^\pi\right)\\&+D^{(t-k-2)}\left(\epsilon_k^{(t-k-1)}+\gamma_{t-1}\boldsymbol{\theta}^\top\bar{\boldsymbol{\phi}}_{t-1}^\pi\right)\\&+\sum_{n=1}^{t-k-2}D_k^{(n-1)}\Big[(1-\gamma_{k+n}\lambda_{k+n})\epsilon_k^{(n)}\\&+\gamma_{k+n}(1-\lambda_{k+n})\boldsymbol{\theta}^\top\bar{\boldsymbol{\phi}}_{k+n}^\pi\Big]\\=&\delta_{k,t-1}^\rho+D_k^{(t-k-1)}\delta_{t-1}+D_k^{(t-k-1)}\boldsymbol{\theta}^\top\left(\boldsymbol{\phi}_{t-1}-\bar{\boldsymbol{\phi}}_{t-1}^\pi\right)\\&+(\rho_{t-1}-1)\gamma_{t-1}\lambda_{t-1}D_k^{(t-k-2)}\left(\epsilon_k^{(t-k-1)}+\boldsymbol{\theta}^\top\bar{\boldsymbol{\phi}}_{t-1}^\pi\right).\end{aligned}$$

The forward-view update at k, given only the data up through S_t, is given as before by:

$$\Delta\boldsymbol{\theta}_{k,t}^F=\alpha\delta_{k,t}^\rho\boldsymbol{\phi}_k. \tag{26}$$

Using the interim equivalence analysis technique, we can derive the backward view as follows:

$$\begin{aligned}&\frac{1}{\alpha}\left(\sum_{k=0}^{t}\Delta\boldsymbol{\theta}_{k,t+1}^F-\sum_{k=0}^{t-1}\Delta\boldsymbol{\theta}_{k,t}^F\right)\\&=\delta_{t,t+1}^\rho\boldsymbol{\phi}_t+\sum_{k=0}^{t-1}\left[\delta_{k,t+1}^\rho-\delta_{k,t}^\rho\right]\boldsymbol{\phi}_k\\&=\delta_t\boldsymbol{\phi}_t+\sum_{k=0}^{t-1}D_k^{(t-k)}\delta_t\boldsymbol{\phi}_k+\sum_{k=0}^{t-1}D_k^{(t-k)}\boldsymbol{\theta}^\top\left(\boldsymbol{\phi}_t-\bar{\boldsymbol{\phi}}_t^\pi\right)\boldsymbol{\phi}_k\\&\quad+(\rho_t-1)\gamma_t\lambda_t\sum_{k=0}^{t-1}D_k^{(t-k-1)}(\epsilon_k^{(t-k)}+\boldsymbol{\theta}^\top\bar{\boldsymbol{\phi}}_t^\pi)\boldsymbol{\phi}_k\\&=\delta_t\boldsymbol{e}_t+\boldsymbol{\theta}^\top\left(\boldsymbol{\phi}_t-\bar{\boldsymbol{\phi}}_t^\pi\right)(\boldsymbol{e}_t-\boldsymbol{\phi}_t)+(\rho_t-1)\boldsymbol{u}_t.\end{aligned}$$

where we define $\boldsymbol{e}_t$ as:

$$\boldsymbol{e}_t=\boldsymbol{\phi}_t+\sum_{k=0}^{t-1}D_k^{(t-k)}\boldsymbol{\phi}_k=\boldsymbol{\phi}_t+\gamma_t\lambda_t\rho_t\boldsymbol{e}_{t-1},$$

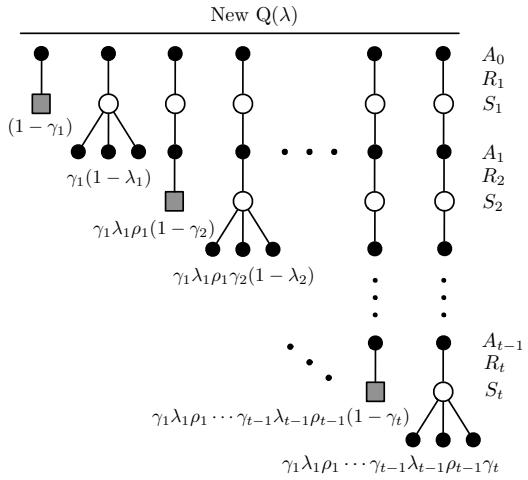

Figure 3. Weighting of Q(λ) backups

and $\boldsymbol{u}_t$ as

$$\begin{aligned}
\boldsymbol{u}_t &= \gamma_t\lambda_t \sum_{k=0}^{t-1} D_k^{(t-k-1)} \left(\epsilon_k^{(t-k)} + \boldsymbol{\theta}^\top \bar{\boldsymbol{\phi}}_t^\pi\right) \boldsymbol{\phi}_k \\
&= \gamma_t\lambda_t \Big(\rho_{t-1}\gamma_{t-1}\lambda_{t-1} \sum_{k=0}^{t-2} D_k^{(t-k-2)} \Big(\epsilon_k^{(t-k-1)} \\
&\quad + \boldsymbol{\theta}^\top \bar{\boldsymbol{\phi}}_{t-1}^\pi\Big)\boldsymbol{\phi}_k + \left(R_t + \boldsymbol{\theta}^\top \bar{\boldsymbol{\phi}}_t^\pi - \boldsymbol{\theta}^\top \boldsymbol{\phi}_{t-1}\right) \boldsymbol{\phi}_{t-1} \\
&\quad + \sum_{k=0}^{t-2} D_k^{(t-k-1)} \left(R_t + \boldsymbol{\theta}^\top \bar{\boldsymbol{\phi}}_t^\pi - \boldsymbol{\theta}^\top \bar{\boldsymbol{\phi}}_{t-1}^\pi\right) \boldsymbol{\phi}_k \Big) \\
&= \gamma_t\lambda_t \Big(\rho_{t-1}\boldsymbol{u}_{t-1} + (R_t + \boldsymbol{\theta}^\top \bar{\boldsymbol{\phi}}_t^\pi - \boldsymbol{\theta}^\top \boldsymbol{\phi}_{t-1})\boldsymbol{e}_{t-1} \\
&\quad + \boldsymbol{\theta}^\top \left(\boldsymbol{\phi}_{t-1} - \bar{\boldsymbol{\phi}}_{t-1}^\pi\right) \left(\boldsymbol{e}_{t-1} - \boldsymbol{\phi}_{t-1}\right)\Big).
\end{aligned}$$

We redefine $\boldsymbol{u}_t = \alpha \boldsymbol{u}_t$. From this backward view, a new off-policy Q(λ) algorithm can be obtained. For any time t, the incremental updates of the new off-policy Q(λ) are:

$$\delta_t = R_{t+1} + \gamma_{t+1}\boldsymbol{\theta}^\top \bar{\boldsymbol{\phi}}_{t+1}^\pi - \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_t \tag{27}$$

$$\boldsymbol{e}_t = \boldsymbol{\phi}_t + \gamma_t\lambda_t\rho_t \boldsymbol{e}_{t-1} \tag{28}$$

$$\Delta\boldsymbol{\theta}_t = \alpha\delta_t \boldsymbol{e}_t + \alpha\boldsymbol{\theta}^\top(\boldsymbol{\phi}_t - \bar{\boldsymbol{\phi}}_t^\pi)(\boldsymbol{e}_t - \boldsymbol{\phi}_t) + (\rho_t - 1)\boldsymbol{u}_t \tag{29}$$

$$\begin{aligned}
\boldsymbol{u}_{t+1} = \gamma_{t+1}\lambda_{t+1} \Big(&\rho_t \boldsymbol{u}_t + \alpha(R_{t+1} + \boldsymbol{\theta}^\top \bar{\boldsymbol{\phi}}_{t+1}^\pi - \boldsymbol{\theta}^\top \boldsymbol{\phi}_t)\boldsymbol{e}_t \\
&+ \alpha\boldsymbol{\theta}^\top(\boldsymbol{\phi}_t - \bar{\boldsymbol{\phi}}_t^\pi)(\boldsymbol{e}_t - \boldsymbol{\phi}_t)\Big).
\end{aligned} \tag{30}$$

Theorem 5. *The backward-view algorithm defined by (27), (28), (29) and (30) is equivalent to the forward view defined in (26).*

The proof immediately follows from Theorem 3 and the above derivation.

In the on-policy case, $\rho = 1$ eliminates the last term in the $\Delta\theta$ update and results in the usual eligibility trace algorithm for action values. On the other hand, $\lambda = 0$ always eliminates the second term, because $\boldsymbol{e}_t = \boldsymbol{\phi}_t$ in that case. As $b_t = \pi_t$ in the on-policy case, the remaining terms in the $\boldsymbol{\theta}$ update coincide with the update of the Expected Sarsa algorithm. With $\lambda > 0$, this algorithm becomes the *summation Q(λ)* by Rummery (1994). In addition, we think this is also a proper implementation of the Expected Sarsa(λ) algorithm in the sense that it has a corresponding equivalent forward-view.

For the control case, if the target policy is greedy wrt the action values, then $\bar{\boldsymbol{\phi}}_t^\pi = \boldsymbol{\phi}(s_t, a^*)$, where $a^* = \operatorname{argmax}_a \boldsymbol{\theta}^\top \boldsymbol{\phi}(s_t, a)$ and $\boldsymbol{\theta}^\top \bar{\boldsymbol{\phi}}_{t+1}^\pi = \max_a \boldsymbol{\theta}^\top \boldsymbol{\phi}(s_t, a)$. Note that the second term in the $\boldsymbol{\theta}$ update is eliminated in this case, because if the action at time t is greedy, then $\boldsymbol{\phi}_t = \bar{\boldsymbol{\phi}}_t^\pi$, and if the action is non-greedy, then $\rho_t = 0$ and hence $\boldsymbol{e}_t = \boldsymbol{\phi}_i$. In that case, the $\boldsymbol{\theta}$ update becomes similar to that of Watkin's Q(λ) except that the eligibility trace involves ρ_t, and there is an additional term $(\rho_t - 1)\boldsymbol{u}_t$. This term is responsible for unmaking the right update when a non-greedy action is taken, which is not done in Q(λ) or any other off-policy control algorithms with eligibility traces. Also note that, the terms in the $\boldsymbol{\theta}$ update can be rearrange in the following way:

$$\begin{aligned}
\Delta\boldsymbol{\theta}_t &= \alpha\delta_t\boldsymbol{\phi}_t + (\rho_t - 1)\boldsymbol{u}_t \\
&\quad + \alpha\left(R_{t+1} + \gamma_{t+1}\boldsymbol{\theta}^\top \bar{\boldsymbol{\phi}}_{t+1}^\pi - \boldsymbol{\theta}^\top \bar{\boldsymbol{\phi}}_t^\pi\right)(\boldsymbol{e}_t - \boldsymbol{\phi}_t).
\end{aligned}$$

Now, if we drop all the ρ_t from the eligibility trace update and drop the $(\rho_t - 1)\boldsymbol{u}_t$ term from the $\boldsymbol{\theta}$ update, then this algorithm becomes Peng's Q(λ). This similarity can also be appreciated from the resemblance in backup diagrams of both algorithms (for a backup diagram of Peng's Q(λ), see Sutton & Barto 1998). The main difference is that our new Q(λ) algorithm weights the component backups with important sampling ratios.

7. Conclusion and future work

We presented a new approach for deriving off-policy eligibility trace TD algorithms, which is general and yields new versions of both TD and Q-learning. The current presentation has focused on the case of fixed target policy and linear function approximation. Convergence in the tabular case should follow by simple application of existing theoretical results. However, when linear function approximation is used a gradient-based extension of these algorithms may be required (cf. Maei 2011), but this extension seems straightforward. Convergence in the control case has never been established for any algorithms for $\lambda > 0$ and this remains an open problem. We presented algorithms which are exactly equivalent in the off-line case, when updates to the parameter vector are applied at the end of a batch of data. Equivalence to online updating, when $\boldsymbol{\theta}$ is modified at every time step, remains to be established. We note that this result has not yet been provided even for usual TD(λ).

References

Maei, H. R., Sutton, R. S. (2010). GQ(λ): A general gradient algorithm for temporal- difference prediction learning with eligibility traces. In *Proceedings of the Third Conference on Artificial General Intelligence*, pp. 91–96. Atlantis Press.

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

Peng, J., Williams, R. J. (1994). Incremental multi-step Q-learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 226–232.

Precup, D., Sutton, R. S., Singh, S. (2000). Eligibility traces for off-policy policy evaluation. In *Proceedings of the 17th International Conference on Machine Learning*, pp. 759–766. Morgan Kaufmann.

Precup, D., Sutton, R. S., Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. In *Proceedings of the 18th International Conference on Machine Learning*, pp. 417–424.

Rummery, G. A. (1995). *Problem Solving with Reinforcement Learning*. PhD thesis, Cambridge University.

Singh, S. P., Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22: 123–158.

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

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

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, Cambridge University.

Watkins, C. J. C. H., Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292.