Replacing eligibility trace for action-value learning with function approximation

Kary FRÄMLING

Helsinki University of Technology
PL 5500, FI-02015 TKK - Finland

Abstract. The eligibility trace is one of the most used mechanisms to speed up reinforcement learning. Earlier reported experiments seem to indicate that replacing eligibility traces would perform better than accumulating eligibility traces. However, replacing traces are currently not applicable when using function approximation methods where states are not represented uniquely by binary values. This paper proposes two modifications to replacing traces that overcome this limitation. Experimental results from the Mountain-Car task indicate that the new replacing traces outperform both the accumulating and the 'ordinary' replacing traces.

1 Introduction

Eligibility traces are an essential element of reinforcement learning because they can speed up learning significantly, especially in tasks with delayed reward. Two alternative implementations of eligibility traces exist, i.e. the *accumulating eligibility trace* and the *replacing eligibility trace*. As reported in [4], the replacing trace would seem to perform better than the accumulating trace. Unfortunately, current replacing traces can not be used with continuous-valued function approximation methods for action-value learning because traces of other states and actions than the intended ones may be affected [5, p. 212].

In this paper, we propose two modifications to replacing eligibility traces that make it possible to use them with continuous-valued function approximation. The modified replacing eligibility trace is compared with accumulating and 'ordinary' replacing traces. The experimental task used is the same as in [4], i.e. the well-known *Mountain-Car* task. Experiments are performed using a grid discretisation (i.e. lookup-table), a CMAC discretisation and a Radial Basis Function (RBF) function approximator.

After this introduction, section 2 describes the background and theory of eligibility traces both for the case of discrete state spaces and for continuous state spaces, including the use of function approximators. Section 2 also explains the modifications to replacing eligibility traces proposed in this paper, followed by experimental results in section 3 and conclusions.

2 Eligibility traces for action-value learning

Most current RL methods are based on the notion of value functions. Value functions either represent state-values (i.e. value of a state) or action-values (i.e.

the value of taking an action in a given state, usually called *Q-value*). Action-value learning is typically needed in control tasks. The SARSA method [3] seems to be the most popular method for learning Q-values for the moment. SARSA is a so-called on-policy method, which makes it easier to handle eligibility traces than for off-policy methods such as Q-learning. In SARSA, Q-values are updated according to

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha \delta_t e_t(s,a), \quad \texttt{for all}\, s, a, \tag{1}$$

where α is a learning rate, $e_t(s,a)$ is the eligibility trace of state s and action a and

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t), \tag{2}$$

where r_{t+1} is the reward received upon entering a new state, γ is the *discount rate* and $Q_t(s_t, a_t)$ and $Q_t(s_{t+1}, a_{t+1})$ are the action-value estimates for the current and next state, respectively.

Eligibility traces are inspired from the behaviour of biological neurons that reach maximum eligibility for learning a short time after their activation. Eligibility traces were mentioned in the context of Machine Learning at least as early as 1972 and used for action-value learning at least as early as 1983 [1]. The oldest version of the eligibility trace for action-value learning seems to be the accumulating eligibility trace:

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & \texttt{if}\ s = s_t\ \texttt{and}\ a = a_t; \\ \gamma\lambda e_{t-1}(s,a) & \texttt{otherwise}. \end{cases} \quad \texttt{for all}\ s, a, \tag{3}$$

where λ is a trace decay parameter. In [4] it was proposed to use a replacing eligibility trace instead of the accumulating eligibility trace:

$$e_t(s,a) = \begin{cases} 1 & \texttt{if}\ s = s_t\ \texttt{and}\ a = a_t; \\ 0 & \texttt{if}\ s = s_t\ \texttt{and}\ a \neq a_t; \\ \gamma\lambda e_{t-1}(s,a) & \texttt{if}\ s \neq s_t. \end{cases} \quad \texttt{for all}\ s, a. \tag{4}$$

The replacing eligibility trace outperformed the accumulating eligibility trace at least in the well-known Mountain-Car task [4], which has also been selected for the experiments reported in section 3. In the Mountain-Car task the state space is continuous and two-dimensional. In order to use equations 1 to 4 with a continuous-valued state space, it becomes necessary to encode the real state vector $\vec{s}$ into a corresponding binary *feature vector*, $\vec{\phi}$. A great amount of work on RL still uses binary features, as obtained by grid discretisation or the CMAC tile-coding [5]. However, it is also possible to perform action-value learning with continuous-valued features or other function approximation methods. In that case, equation 1 becomes [5, p. 211]:

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha \left[v_t - Q_t(s_t, a_t) \right] \nabla_{\vec{\theta}_t} Q_t(s_t, a_t), \tag{5}$$

where $\vec{\theta}_t$ is the parameter vector of the function approximator. For SARSA, the target output $v_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1})$, which gives the update rule:

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha \delta_t \vec{e}_t \texttt{, where} \tag{6}$$

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \texttt{, and} \tag{7}$$

$$\vec{e}_t = \gamma \lambda \vec{e}_{t-1} + \nabla_{\vec{\theta}_t} Q_t(s_t, a_t) \tag{8}$$

with $\vec{e}_0 = \vec{0}$, which represents an accumulating eligibility trace. This method is called *gradient-descent Sarsa(λ)* [5, p. 211]. Equation 3 is a special case of equation 8 when using binary state representations because then $\nabla_{\vec{\theta}_t} Q_t(s_t, a_t) = 1$ when $s = s_t$ and $a = a_t$ and zero otherwise. When artificial neural networks (ANNs) are used for learning an action-value function, an eligibility trace value is usually associated with every weight of the ANN [1]. Then the eligibility trace value reflects to what extent and how long ago the neurons connected by the weight have been activated.

It seems like no similar generalisation exists for the replacing trace in equation 4. In this paper, we propose the following update rule for replacing traces:

$$\vec{e}_t = \max(\gamma \lambda \vec{e}_{t-1}, \nabla_{\vec{\theta}_t} Q_t(s_t, a_t)). \tag{9}$$

Even though no theoretical or mathematical proof is proposed here for this update rule, it makes it possible to compare the performance of accumulating and replacing eligibility traces also for continuous function approximation. Except for the resetting of unused actions, Equation 4 is a special case of equation 9 when using binary state representations. In addition to the resetting of unused actions, setting an action's trace to one for the used action in Equation 4 is a major obstacle for using replacing traces with continuous-valued function approximators. In the case of binary features, this is feasible because the traces to set or reset can be uniquely identified. For continuous-valued features, it would be necessary to define thresholds for deciding if a feature is sufficiently 'present' for representing the current state or not in order to set the corresponding trace values to one or to zero. Furthermore, there seems to be no strong theoretical foundation for resetting the traces of unused actions to zero. In [4] it is indicated that resetting the trace was considered preferable due to its similarity with first-visit Monte-Carlo methods, while it is indicated as optional in [5]. Both [4] and [5] also call for empirical evidence on the effects of resetting the traces of unused actions but it seems like no such experimental results have been reported yet. The experimental results in the next section will attempt to give more insight into this question.

3 Mountain-Car experiments

In order to simplify comparison with earlier work on eligibility traces in control tasks, we will here use the same Mountain-Car task as in [4] where the dynamics of the task are described. The task has a continuous-valued state vector $\vec{s}_t = (position, velocity)$ of the car. The task consists in accelerating an under-powered car up a hill, which is only possible if the car first gains enough inertia by backing away from the hill. The three possible actions are full throttle forward (+1), full throttle backward (-1) and no throttle (0). The reward function used is the *cost-to-go* function as in [4], i.e. giving -1 reward for every step except when reaching the goal, where zero reward is given. The parameters $\vec{\theta}$ were set to zero before every new run so the initial action-value estimates are zero. Therefore the initial action-value estimates are optimistic compared to the actual action-value function, which encourages exploration of the state space. The discount rate γ was set to one, i.e. no discounting.

In addition to the CMAC experiments used in [4], experiments were performed using a grid discretisation ('lookup-table') and a continuous-valued Radial Basis Function (RBF) function approximator. The grid discretisation used 8×8 tiles while CMAC used five 9×9 tilings as in [4]. CMAC tilings usually have a random offset in the range of one tile but for reasons of repeatability a regular offset of $\frac{tile-range}{number-of-tilings}$ was used here. With the RBF network, feature values are calculated as:

$$\phi_i(\vec{s}_t) = \exp\left(-\frac{(\vec{s}_t - \vec{c})^2}{r^2}\right), \tag{10}$$

where $\vec{c}$ is the *centroid* vector of the RBF neuron and r is the *spread* parameter. 8×8 RBF neurons were used with regularly spaced centroids. In the experiments, an affine transformation mapped the actual state values from the intervals [-1.2,0.5] and [-0.07,0.07] into the interval [0,1] that was used as $\vec{s}$. This transformation makes it easier to adjust the spread parameter ($r^2 = 0.01$ used in all experiments). The action-value estimate of action a is then:

$$Q_t(\vec{s}_t, a) = \sum_{i=1}^{N} w_{ia}\left(\phi_i / \sum_{k=1}^{N} \phi_k\right), \tag{11}$$

where w_{ia} is the weight between the 'action' neuron a and the RBF neuron i. The division by $\sum_{k=1}^{N} \phi_k$, where N is the number of RBF neurons, implements a *normalized RBF* network. Equation 11 without normalisation was also used for the binary features produced by grid discretisation and CMAC. Every parameter w_{ia}, has its corresponding eligibility trace value. For the accumulating trace, equation 8 was used in all experiments, while equation 9 was used for the replacing trace.

Results for the three methods *accumulating* trace, replacing trace *with reset* for unused actions and replacing trace *without reset* for unused actions are compared in Figure 1. The results are indicated as the average number of steps per episode for 50 agents that each performed 200 episodes for grid discretisation and 100 episodes for CMAC and RBF. Figure 1 shows the results obtained with the best learning rate values found, which are indicated in Table 1. Every episode started with a random position and velocity.

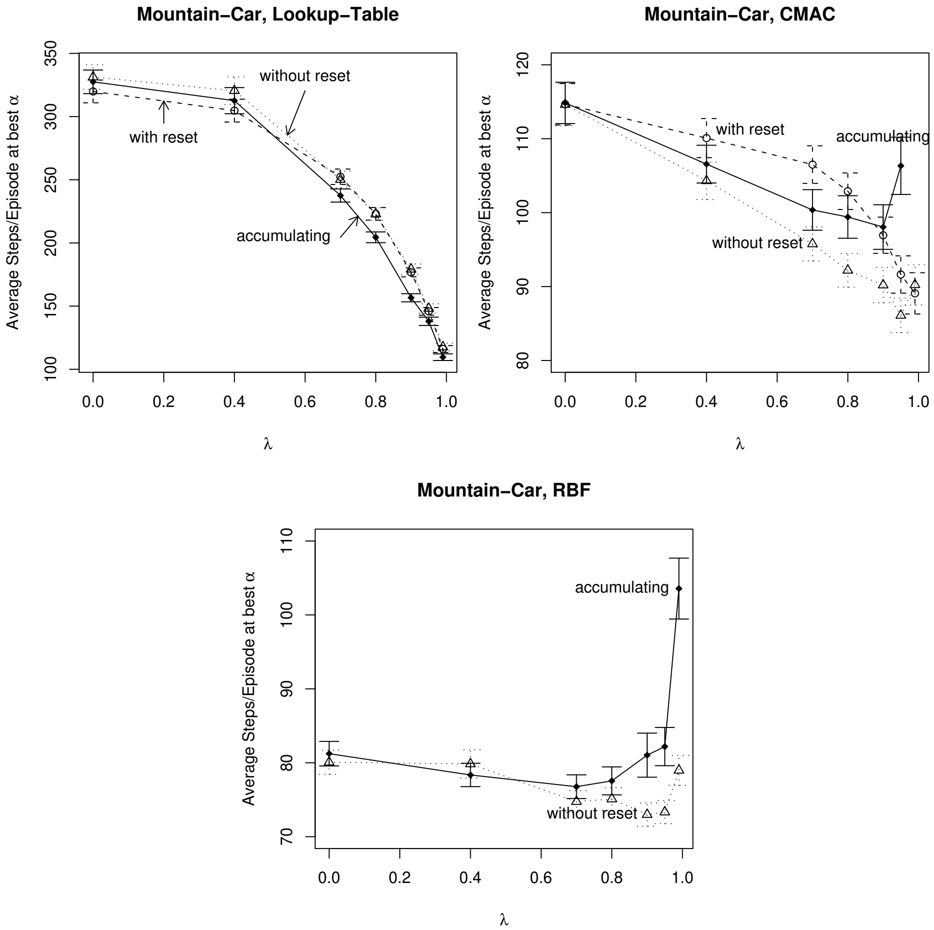

Fig. 1: Results with best α value of different methods as a function of λ. Error bars indicate one standard error.

There is no significant difference between the three methods when using grid discretisation. With CMAC, the replacing trace without reset is the best while the accumulating trace performs the worst. With RBF, the replacing trace with reset is not applicable due to the reasons explained in section 2. The replacing trace without reset outperforms the accumulating trace. As indicated by Table 1, the accumulating trace is more sensitive to the value of the learning rate than the replacing traces. This is because the $\vec{e}_t$ value in Equation 6 becomes bigger for the accumulating trace than for the replacing traces so the learning rate needs to be decreased correspondingly. Better results might be obtained by fine-tuning of the learning rate or by using a normalized least mean square (NLMS) version of Equation 6 (see e.g. [2] on the use of NLMS in a reinforcement learning task).

Method		λ=0	λ=0.4	λ=0.7	λ=0.8	λ=0.9	λ=0.95	λ=0.99
Accumulating	Grid	0.3	0.3	0.1	0.1	0.1	0.05	0.05
	CMAC	0.26	0.2	0.14	0.1	0.08	0.04	NA
	RBF	4.7	3.3	2.3	2.1	0.9	0.5	0.1
With reset	Grid	0.3	0.3	0.3	0.3	0.3	0.3	0.3
	CMAC	0.28	0.24	0.28	0.3	0.24	0.22	0.12
	RBF	NA	NA	NA	NA	NA	NA	NA
Without reset	Grid	0.3	0.3	0.3	0.3	0.3	0.3	0.3
	CMAC	0.26	0.26	0.28	0.26	0.2	0.22	0.14
	RBF	4.9	4.3	4.3	4.7	3.9	3.3	2.5

Table 1: Best α values for different values of λ. NA: Not Applicable.

4 Conclusions

The experimental results confirm that replacing traces give better results than accumulating traces with CMAC for the Mountain-Car task. Replacing traces are also more stable against changes in learning parameters than accumulating traces. The replacing eligibility trace without reset of unused actions gives the best results with both CMAC and RBF. Therefore, the replacing eligibility trace proposed in this paper (Equation 9) is the best choice at least for this task. Experiments with other tasks should still be performed to see if the same applies to them.

It also seems that *structural credit assignment* (i.e. weights of function approximator in this case) is more powerful than *temporal credit assignment* (i.e. eligibility traces in this case). The improvement is greater when passing from grid discretisation to CMAC to RBF than what can be obtained by modifying the eligibility trace or its parameters. For instance, the best result with RBF and no eligibility trace is 80 steps/episode, which is better than the best CMAC with $\lambda = 0.95$ (86 steps/episode). Therefore the utility of eligibility traces seems to decrease if appropriate structural credit assignment can be provided instead.

References

[1] A.G. Barto, R.S. Sutton and C.W. Anderson, Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems, *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835-846, 1983.

[2] K. Främling, Adaptive robot learning in a non-stationary environment. In M. Verleysen, editor, *proceedings of the* 13^{th} *European Symposium on Artificial Neural Networks* (ESANN 2005), pages 381-386, April 27-29, Bruges (Belgium), 2005.

[3] G.A. Rummery and M. Niranjan, On-Line Q-Learning Using Connectionist Systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994.

[4] S.P. Singh and R.S. Sutton, Reinforcement Learning with Replacing Eligibility Traces, *Machine Learning*, 22:123-158, 1996.

[5] R.S Sutton and A.G. Barto. *Reinforcement Learning*, MIT Press, Cambridge, Massachusetts, 1998.