Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Greedy feature replacement for online value function approximation*

Feng-fei ZHAO†1, Zheng QIN†1,2, Zhuo SHAO†3, Jun FANG1, Bo-yan REN2

(1*Department of Computer Science and Technology*, *Tsinghua University*, *Beijing 100084*, *China*)

(2*School of Software*, *Tsinghua University*, *Beijing 100084*, *China*)

(3*Department of Physics and State Key Laboratory of Low-Dimensional Quantum Physics*, *Tsinghua University*, *Beijing 100084*, *China*)

†E-mail: zhaofengfei@gmail.com; qingzh@tsinghua.edu.cn; shaoz09@mails.tsinghua.edu.cn

Received Sept. 6, 2013; Revision accepted Jan. 14, 2014; Crosschecked Feb. 21, 2014

Abstract: Reinforcement learning (RL) in real-world problems requires function approximations that depend on selecting the appropriate feature representations. Representational expansion techniques can make linear approximators represent value functions more effectively; however, most of these techniques function well only for low dimensional problems. In this paper, we present the greedy feature replacement (GFR), a novel online expansion technique, for value-based RL algorithms that use binary features. Given a simple initial representation, the feature representation is expanded incrementally. New feature dependencies are added automatically to the current representation and conjunctive features are used to replace current features greedily. The virtual temporal difference (TD) error is recorded for each conjunctive feature to judge whether the replacement can improve the approximation. Correctness guarantees and computational complexity analysis are provided for GFR. Experimental results in two domains show that GFR achieves much faster learning and has the capability to handle large-scale problems.

Key words: Reinforcement learning, Function approximation, Feature dependency, Online expansion, Feature replacement
doi:10.1631/jzus.C1300246 **Document code:** A **CLC number:** TP181

1 Introduction

The complex, high dimensional problem is the main obstacle in the transition of reinforcement learning (RL) from the toy world to real-world applications. Traditional tabular reinforcement learning has high demands on computing time and storage space, making RL in large spaces a challenge (Tsitsiklis, 1994; Sprague and Ballard, 2003). However, using function approximation introduces generalization into RL, effectively reducing time and space burdens, and providing a solution to high dimensional problems (Singh and Yee, 1994; Sutton, 1996).

Linear value function approximation is widely considered an effective method for RL owing to its simple form and complete theoretical basis. Nevertheless, linear approximators using only initial features run poorly in many applications, as they cannot capture the relationship between features to express the value function well. Researchers such as Albus (1971) and Sturtevant and White (2006) have pursued manual approaches that recognize the important feature dependencies to improve the learning performance; however, discovering feature dependencies automatically is clearly more attractive. Several online automatic expansion techniques have been developed, such as adaptive tile coding (ATC) and sparse distributed memories (SDM) (Ratitch and Precup, 2004; Whiteson *et al*., 2007), which simplify the learning process. However, these techniques still cannot handle large-scale problems well, because their new features must correspond to the full

* Project supported by the 12th Five-Year Defense Exploration Project of China (No. 041202005) and the Ph.D. Program Foundation of the Ministry of Education of China (No. 20120002130007)

dimensional state space even though a lower dimensional space is enough.

The recently proposed online expansion technique, incremental feature dependency discovery (iFDD), incrementally applies feature dependencies by defining new features in low dimensional subspaces of the whole state space (Geramifard *et al.*, 2011). Whether iFDD adds a new feature dependency to feature representation depends on the accumulation of corresponding temporal difference (TD) errors. When the accumulation exceeds a predetermined threshold, a new feature dependency will be added. iFDD can be applied to large space problems; however, as iFDD uses only the sum of the absolute values of the approximation errors to determine whether to add a feature dependency, some important feature dependencies cannot be identified timely.

In this work, we present greedy feature replacement (GFR) as a new online feature expansion technique. GFR has the ability to tackle large-scale problems, and can be applied to any online, value-based method. GFR introduces a variable, the virtual TD error, to simulate the TD error after a new feature has been added to the feature representation. GFR identifies feature conjunctions by analyzing the differences between the actual and virtual TD errors. If the differences exceed a predetermined threshold, the conjunctive feature will replace the original features. In each replacement, two original features are replaced by one conjunctive feature. GFR can optimize the sequence in which the feature conjunctions are added, thus improving the performance of representational expansion. We provide correctness guarantees and computational complexity analysis for GFR and show the superiority of GFR through two experiments. Compared to other online expansion techniques, GFR can learn much faster and has advantages in solving large-scale problems.

2 Background

2.1 Markov decision process

The Markov decision process (MDP) is the theoretical basis of RL (Puterman, 1994; Barto *et al.*, 1995; Kolter and Ng, 2009; Pazis and Lagoudakis, 2009). An MDP is formalized as a tuple $M=(S, A, P, R, \gamma, D)$, where S is the set of possible states, A is the set of possible actions, P denotes the state transition probabilities—$p(s, a, s')$ is the transition probability of taking action a from state s to s', R is the corresponding reward function—$r(s, a)$ denotes the expected reward for taking action a in state s, γ is a real number between 0 and 1, which is the discount factor for future rewards, and D represents the initial state distribution. A policy π: $S\rightarrow A$ is a mapping from the states to the actions, and $\pi(s)$ gives the action selection in state s.

2.2 Reinforcement learning

As an important research branch of machine learning, RL improves performance through interaction with the environment (Kaelbling *et al.*, 1996; Sutton and Barto, 1998; de Hauwere *et al.*, 2010). RL creates a mapping from the environment states to the actions so that the maximum cumulative reward can be obtained. Different from supervised learning techniques, RL discovers the best policy through repeated attempts, and does not require tagged training samples.

In the learning process, r_t is the reward for an agent starting at state s_t, executing action a_t, and ending at s_{t+1}. The above step is repeated and a series of states, actions, and rewards are generated. The target of the learning agent is to learn a policy π, which can maximize the sum of all these discounted rewards: $r_0+\gamma r_1+\gamma^2 r_2+\dots$, $0\le\gamma<1$. We define the value function as $V^{\pi}(s)=E_{\pi}\left\{\sum_{i=0}^{\infty}\gamma^i r_{t+i} \mid s_t=s\right\}$, and then the optimal policy can be expressed as

$$\pi^*=\arg\max_{\pi} V^{\pi}(s) \quad \forall s. \tag{1}$$

According to dynamic programming theory (Puterman, 1994), the optimal policy in state s can be written as

$$\pi^*(s)=\arg\max_{a} E_{\pi^*}\{r(s,a)+\gamma V^*(s')\}, \tag{2}$$

where s' represents the successor state and is drawn from the distribution $P(s, a, s')$, and V^* denotes the optimal value function. In order to represent the greedy policy without the need to store the policy explicitly, the Q function which associates values with state–action pairs is introduced:

$$Q(s,a) = E\{r(s,a) + \gamma V^*(s')\}. \qquad (3)$$

2.3 SARSA learning

SARSA is an on-policy, model-free RL algorithm which learns a policy directly, instead of by modeling the environment (Rummery and Niranjan, 1994; Singh and Sutton, 1996). Different from *Q*-learning (Watkins and Dayan, 1992), the behavior policy and the target policy are consistent in SARSA. In *Q*-learning, the value function is updated based on the maximum *Q* value of the successor state, while SARSA uses the *Q* value of the actual action taken by the learner (Singh *et al.*, 2000).

SARSA updates the *Q* values according to the following rule:

$$Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \alpha[r_t + \gamma Q_t(s',a') - Q_t(s,a)], \qquad (4)$$

where α is the learning rate, and s' and a' represent the next state and action, respectively. The optimal policy for SARSA can be rewritten as

$$\pi^*(s) = \arg\max_a Q(s,a). \qquad (5)$$

2.4 Linear function approximation

Tabular RL methods such as *Q*-learning and SARSA have complete theoretical bases and convergence proofs. However, these tabular methods are not able to handle high dimensional problems because of the lack of generalization (i.e., sharing information among similar states). The common practice of tackling this problem is to use a linear approximation in the form of $Q(s, a)=\boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\phi}(s, a)$, where $\boldsymbol{\phi}$ is the feature vector generated from s and a, and $\boldsymbol{\theta}$ is the weight vector (Buro, 1999). The learning process of RL with value function approximation is actually the process of adjusting $\boldsymbol{\theta}$. The rule for updating $\boldsymbol{\theta}$ can be written as

$$\boldsymbol{\theta}_{t+1}=\boldsymbol{\theta}_t+\alpha\delta_t\boldsymbol{\phi}(s, a), \qquad (6)$$

where δ_t is the TD error:

$$\delta_t=r_t+\gamma Q_t(s', a')-Q_t(s, a). \qquad (7)$$

3 Greedy feature replacement

3.1 Method

We employ linear value function approximation with binary features. Using binary feature representation, computing the estimated values is simple and fast; therefore, it has been widely used in RL (Albus, 1971; Sutton, 1996). In binary representation, the conjunction of two features can represent the dependency between them. By adding feature conjunctions to the representation, nonlinearities can be captured by the linear value function approximator. Therefore, a linear approximator can achieve the effect of a nonlinear one that uses these initial features (Geramifard *et al.*, 2011).

Like iFDD, GFR selects feature conjunctions as new features and incrementally adds them to the feature representation. Therefore, the feature conjunction selection process is the key point of our algorithm. The GFR algorithm is based on a simple principle: If the feature representation has been perfect, adding new feature conjunctions will not further reduce the approximation error. Namely, if adding new feature conjunctions can reduce the approximation error, the feature representation still needs to be expanded. Different from iFDD, GFR can optimize the sequence in which the feature conjunctions are discovered and improve the performance of representational expansion.

In the GFR algorithm, we expand the feature representation in the form of feature replacement. By using $\phi_f(s, a)=1$ to indicate that the binary feature f in the current feature set is true (also called ‘active’) when taking action a in state s, we derive the definition of the term ‘feature replacement’.

Definition 1 (Feature replacement) Given two features d and e in the current feature representation, the feature replacement of d and e is an incremental local expansion which is to: (1) add $c=d\wedge e$ to the feature representation; (2) set $\phi_c(s, a)=1$ and $\phi_d(s, a)=\phi_e(s, a)=0$ any time both d and e are active.

Now selecting appropriate features to perform feature replacement is the task of our algorithm. GFR simulates a virtual TD error (VE) calculating the TD error as if the new conjunctive feature had been added to the feature representation. The VE only approximately calculates the TD error after feature replacement. In fact, the addition of the feature would also affect the values of the states where the conjunctive feature is not active, which is not considered by the VE. The resulting TD error in the region where the conjunctive feature is active might therefore be different from the value of its VE. We write the VE value corresponding to features d and e at time step t as $\upsilon_t(d\wedge e)$. We term the features that need to be replaced as the 'replacing area'. The formal definition for the replacing area is as follows:

Definition 2 (Replacing area) The feature conjunction of d and e is a replacing area at time step t if and only if: given a threshold $p\geq 0$ and a fixed $\beta\in(0, 1)$, we have $\sum_{i=0}^{t}(|\delta_i|-\beta^i|\upsilon_i(d\wedge e)|)\eta_i > p$, where $\eta_i=[\phi_d(s_i, a_i)=1][\phi_e(s_i, a_i)=1]$ and $[\cdot]$ stands for the Iverson bracket.

In Definition 2, β is necessary for theoretical results and usually close to 1. One important reason for replacing the area is that the change trend of the weights when both features are active is inconsistent with the trend when only one feature is active. For instance, we consider the following situation: there are two features d and e in the representation and the TD error is negative when both of d and e are active, while if only d is active, the TD error is positive. According to Eq. (6), a positive δ_t will increase the weight of feature d, and a negative one will lead to a weight decrease. The positive and negative adjustments are mutually offset continuously, so the weight of d is changing back and forth and the approximation errors in sub-regions cannot be further reduced. This is a conflict between the separate occurrence and joint occurrence, which illustrates the necessity of separating the two cases. By using feature replacement, we expand d and e to three features, d, e, and $d\wedge e$, so that the weight adjustments to the individual features and the combination are separated.

GFR uses a greedy procedure to discover the replacing areas. The GFR algorithm maintains two variables for every simultaneously activated feature pair. One variable is the approximation error sum (AES), which represents the sum of the absolute values of the TD errors when the two features are both active. The other variable is the virtual error sum (VES), which is the weighted sum of the absolute value of VE. Mathematically, $\mathrm{AES}_{d\wedge e}=\sum|\delta_i|\eta_i$ and $\mathrm{VES}_{d\wedge e}=\sum\beta^i|\upsilon_i(d\wedge e)|\eta_i$, where $\beta\in(0, 1)$ and $\eta_i=[\phi_d(s_i, a_i)=1][\phi_e(s_i, a_i)=1]$ as in Definition 2. A threshold p is introduced in order to adapt to the stochasticity of the environment. If the AES and VES of a feature conjunction satisfy AES–VES>p, we regard the feature pair as a replacing area and perform feature replacement. The method that only greedily considers the AES and VES of the current feature conjunction to determine whether to perform feature replacement is referred to as greedy feature replacement (GFR).

As shown in Fig. 1, the circles below represent the features that have been used by the approximator, including the initial features and the newly added ones. The bar chart above depicts the accumulation of approximation errors for each simultaneously activated feature pair, in which the dark bar represents AES and the light one represents VES. If the AES is higher than the VES and the difference exceeds the predetermined threshold, then the corresponding feature conjunction is identified as a replacing area and added to the current feature representation. The above procedure completes a greedy feature replacement. This procedure can be divided into four steps: (1) capture the simultaneously activated feature conjunctions; (2) track the AES and VES of each conjunction; (3) identify the replacing areas; (4) perform feature replacement.

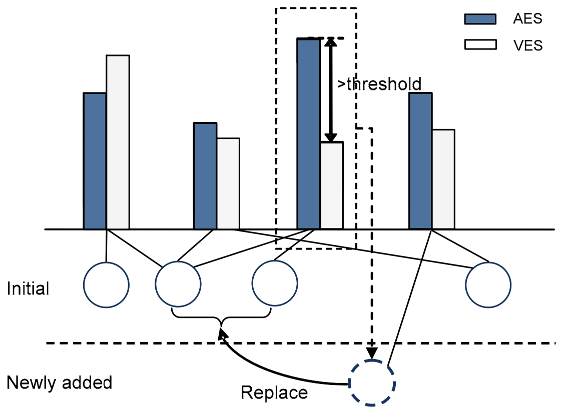

Fig. 1 The principle of greedy feature replacement
Initial features are represented by solid circles and newly added features by dotted circles. The bar chart tracks the accumulation of approximation errors for each simultaneously activated feature pair. If the AES is higher than the VES and the difference reaches the user-defined threshold, feature replacement is performed

3.2 Algorithm details

As mentioned earlier, GFR expands the number of features according to the rule AES–VES>p. Algorithm 1 shows the implementation details of our method. 'Features' represents the current feature set and $\boldsymbol{\phi}(s, a)$ is a binary vector that indicates whether a feature is active when taking action a in state s. At the beginning, 'Features' contains only the initial features. We use the virtual weight (VW) to simulate the weights of the feature conjunctions for all potential replacing areas, as if the feature conjunctions had already been added. We use $\boldsymbol{\omega}$ to represent VW and ω_c the VW value corresponding to feature c.

At the beginning of each time step, we add all conjunctions of active features to the container 'Potential', which can be traversed (line 1). For each conjunction in Potential, if it is not in the Features of the current feature representation, GFR computes its virtual TD error υ_t (lines 6–8):

$$\upsilon_t=r_t+\gamma Q_t(s', a')-Q_t'(s, a), \tag{8}$$

where $Q_t'(s, a)$ is a virtual value which simulates the Q value when the current conjunction has already been added to the representation. According to Eq. (7), we can rewrite Eq. (8) as

$$\upsilon_t=\delta_t+Q_t(s, a)-Q_t'(s, a). \tag{9}$$

Algorithm 1 Greedy feature replacement (GFR)

```
Input: Features, AES, VES, $\boldsymbol{\phi}(s, a)$, $\boldsymbol{\theta}$, $\boldsymbol{\omega}$, $\delta_t$, $p$, $\alpha$, $\beta$.
Output: Features, AES, VES, $\boldsymbol{\omega}$.
1  Potential$\leftarrow\{(i,j)|\phi_i(s, a)\phi_j(s, a)=1\}$
2  while Potential.hasnext() do
3    $(d, e)\leftarrow$Potential.next()
4    $c\leftarrow d\wedge e$
5    if Features.has($c$)=false then
6      weightOfDE$\leftarrow\theta_d+\theta_e$
7      weightOfC$\leftarrow\omega_c$
8      $\upsilon_t(c)\leftarrow\delta_t+$weightOfDE$-$weightOfC
9      AES$_c\leftarrow$AES$_c+|\delta_t|$
10     VES$_c\leftarrow$VES$_c+\beta^t|\upsilon_t(c)|$
11     $\omega_c\leftarrow\omega_c+\alpha\cdot\upsilon_t(c)$
12     if AES$_c-$VES$_c>p$ then
13       Features.add($c$)
14       add $c=d\wedge e$ to ReplaceTable
15     end if
16   end if
17 end while
```

If the feature conjunction in question is $c=d\wedge e$, then the formula that GFR uses to compute $Q_t'(s, a)$ is $Q_t'(s, a)=Q_t(s, a)+\omega_c-\theta_d-\theta_e$. Hence, the VE value for the conjunctive feature c can be calculated by $\upsilon_t(c)=\delta_t+\theta_d+\theta_e-\omega_c$ (line 8). Then the algorithm increases AES and VES by adding the absolute TD error $|\delta_t|$ and the weighted absolute virtual approximation error $\beta^t|\upsilon_t|$, respectively (lines 9 and 10). Similar to the way of updating the weights, GFR updates the virtual weight using υ_t at each time step (line 11). If the difference value (AES–VES) corresponding to the conjunction exceeds the threshold p, GFR regards the conjunction as a replacing area and performs a feature replacement (lines 12–14). Through feature replacement, the tracked conjunction is added to the Features of feature representation and also the ReplaceTable. If $c=d\wedge e$ appears in the ReplaceTable, our algorithm always sets $\phi_c(s, a)=1$ and $\phi_d(s, a)=\phi_e(s, a)=0$ when both d and e are active in the feature vector generated from s and a.

3.3 Computational complexity analysis

Let m_0 be the number of active features at the initial time. The per-step computational complexity of GFR is $O(m_0^2)$, as analyzed in Geramifard *et al.* (2011). Besides the main algorithm, GFR needs to use the current feature representation to cover the active initial features. Geramifard *et al.* (2011) has pointed out that the complexity of approximately computing the minimal cover is $O(m_0 2^{m_0})$. Hence, the actual total per-step complexity of GFR is $O(m_0 2^{m_0})$. Since the number of active features is much smaller than the total number of features, m_0 is usually a small value, which ensures that GFR can run fast.

3.4 Correctness guarantees

Now we shall prove the correctness of the expansion method of GFR combined with TD. Given finite initial features, we argue that GFR can achieve the optimal approximation results theoretically. To prove this, we first prove that either GFR combined with TD will achieve the optimal representation or all possible feature conjunctions will become replacing areas.

Theorem 1 Given a finite set of binary initial features, a discount factor $\gamma\in(0, 1)$, and a policy that can induce an ergodic Markov chain, GFR combined with TD guarantees that all possible feature conjunctions will become replacing areas or that the TD errors are identically zero everywhere with probability one.

Proof We prove this theorem by reductio ad absurdum. Suppose that GFR has reached the final feature representation *F* which neither eliminates the TD error everywhere nor makes all possible feature conjunctions become replacing areas no matter how long the algorithm runs. As GFR does not make all feature conjunctions become replacing areas, there are at least two features *d* and *e* in *F*, and their conjunction $c=d\wedge e$ cannot become a replacing area until the final feature representation. Since the representation cannot set all the TD errors to zero, there exists at least a state *s* in which both *d* and *e* are active and the TD error is nonzero (In an ergodic Markov chain, if there is a state for which the TD error is nonzero, the TD error of the state will propagate to *s* by a finite length path when $\gamma\neq 0$). We rewrite the sum $\sum_{i=0}^{+\infty}(|\delta_i|-\beta^i|\upsilon_i(d\wedge e)|)\eta_i$ in the form of AES–VES: $\sum_{i=0}^{+\infty}|\delta_i|\eta_i-\sum_{i=0}^{+\infty}\beta^i|\upsilon_i(d\wedge e)|\eta_i$. As there might be more than one state in which both *d* and *e* are active, we can obtain a subsequence of $\sum_{i=0}^{+\infty}|\delta_i|\eta_i-\sum_{i=0}^{+\infty}\beta^i|\upsilon_i(d\wedge e)|\eta_i$ for state *s*: $\sum_{i=0}^{+\infty}|\delta_i|\sigma(i,s)-\sum_{i=0}^{+\infty}\beta^i|\upsilon_i(d\wedge e)|\sigma(i,s)$, where $\sigma(i, s)$ indicates whether the state is *s* at time *i*.

Geramifard *et al.* (2011) has proven that the sum $\sum_{i=0}^{+\infty}|\delta_i|\sigma(i,s)$ diverges if the representation is not perfect (Note that if the policy changes, $|\delta_i|$ will not converge to 0 simply because the successor state varies). We now argue that the sum $\sum_{i=0}^{+\infty}\beta^i|\upsilon_i(d\wedge e)|\sigma(i,s)$ has an upper bound. Given a finite set of binary initial features, the value of $|\upsilon_i(d\wedge e)|$ is upper bounded by some constant *h*. Hence, the sum $\sum_{i=0}^{+\infty}\beta^i|\upsilon_i(d\wedge e)|\sigma(i,s)$ has an upper bound $\sum_{i=0}^{+\infty}\beta^i h=h/(1-\beta)$. Therefore, $\sum_{i=0}^{+\infty}|\delta_i|\sigma(i,s)-\sum_{i=0}^{+\infty}\beta^i|\upsilon_i(d\wedge e)|\sigma(i,s)$ diverges. Similarly, the other subsequences of $\sum_{i=0}^{+\infty}(|\delta_i|-\beta^i|\upsilon_i(d\wedge e)|)\eta_i$ diverge. Consequently, $\sum_{i=0}^{+\infty}(|\delta_i|-\beta^i|\upsilon_i(d\wedge e)|)\eta_i$ diverges. However, as $c=d\wedge e$ cannot become a replacing area until the final feature representation, $\sum_{i=0}^{+\infty}(|\delta_i|-\beta^i|\upsilon_i(d\wedge e)|)\eta_i\leq p$. There now exists a contradiction, and thus the original assumption is not true. Hence, Theorem 1 is proven.

As shown in Algorithm 1, GFR performs feature replacement for all the replacing areas and adds the corresponding feature conjunctions to the feature representation, so we can easily obtain the following conclusion:

Corollary 1 Given a finite set of binary initial features, a discount factor $\gamma\in(0, 1)$, and a policy that can induce an ergodic Markov chain, GFR combined with TD is guaranteed to add all possible feature conjunctions or make the TD errors identically zero everywhere with a probability of one.

If GFR sets all the TD errors to zero, GFR apparently achieves the optimal approximation results. The case in which the feature representation is fully expanded has been proven by Tsitsiklis and van Roy (1997) and Geramifard *et al.* (2011). Using the weighted Euclidean norm of the approximated value function error as the quality criterion for value function approximation, Geramifard *et al.* (2011) guaranteed that the fully expanded case (with all possible feature conjunctions added) makes the best possible use of given features. In summary, given the finite initial features, GFR can lead to the optimal approximation results asymptotically.

When the perfect feature representation cannot be obtained, both GFR and iFDD are guaranteed to add all possible feature conjunctions asymptotically. The difference is that GFR can optimize the sequence in which the feature conjunctions are added by identifying the replacing areas. Therefore, GFR improves the performance of representational expansion.

4 Experiments

We chose two representative domains to compare GFR against other feature representation techniques including initial representation, tabular representation, iFDD, and iFDD+ (a new version of iFDD proposed by Geramifard *et al.* (2013)). The ϵ-greedy policy was employed for exploration. We used episodic RL in each domain. For both domains, the following learning rate formula is used:

$$\alpha_t = \frac{\alpha_0}{k_t} \cdot \frac{N_0+1}{N_0+n_t}, \qquad (10)$$

where α_0 and N_0 are the given constants, and k_t and n_t are the numbers of active features and episodes at time step *t*, respectively. The experimental results were the averages of 30 runs. We set the factor $\beta=1-10^{-5}$ in both domains.

4.1 Inverted pendulum

The inverted pendulum domain shown in Fig. 2 is a classic study case that requires balancing a pendulum at the upright position by applying forces to the cart to which it is attached (Lagoudakis and Parr, 2003). The task is to balance the pendulum as long as possible, but the learner does not know the length or the mass of the pendulum. *M* is the mass of the cart, *m* is the mass of the pendulum, and *l* is the length of the pendulum. During each time step, three actions are selectable: left force (−50 N), right force (+50 N), or no force (0 N) (In Geramifard *et al.* (2011), the actions are three fixed torques applied to the pendulum, which induce three fixed angular accelerations). Every action has a random noise *u* from −10 N to 10 N and the noise is uniformly distributed. The state space has two dimensions, which are the vertical angle θ and the angular velocity $\dot{\theta}$ of the pendulum. Initially each dimension is discretized into 20 levels. The angular acceleration $\ddot{\theta}$ of the pendulum is calculated by

$$\ddot{\theta} = \frac{g\sin\theta - \rho ml\dot{\theta}^2\sin(2\theta)/2 - \rho u\cos\theta}{4l/3 - \rho ml\cos^2\theta}, \qquad (11)$$

where *g* is the gravity constant (9.8 m/s^2), $\rho=1/(m+M)$, $l=0.5$ m, $m=2$ kg, and $M=8$ kg. The interval between two time steps is 0.05 s. At each time step, if the absolute value of the angle does not exceed π/2, the learner receives a reward of 0; if greater than π/2, the episode ends with a reward of −1.

The learner learned 1000 episodes for each run. According to the method for parameter selection in Geramifard *et al.* (2011), we set $\alpha_0=0.1$, $N_0=1000$, $\gamma=0.9$, and $\epsilon=0.1$. We found the best threshold $p=0.1$ empirically from $\{0, 0.001, 0.01, 0.1, 1\}$. Fig. 3 shows the average number of balancing steps in one episode (averaged over 30 independent runs) with the standard error of the mean as a function of the number of training episodes. In this domain, the performance of GFR was better than those of all the other methods after 400 episodes. GFR was the only method that can balance the pendulum for more than 2000 steps, which suggests that the feature replacement mechanism is reasonable. The performances of iFDD, iFDD+, and SARSA with tabular representation were almost the same because iFDD and iFDD+ cannot show their advantage of value function approximation in the relatively low dimensional domain. Using the initial representation, the learner never obtained satisfactory results because it cannot approximate the value function well without adding feature conjunctions.

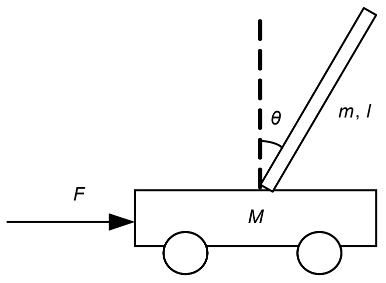

Fig. 2 Inverted pendulum

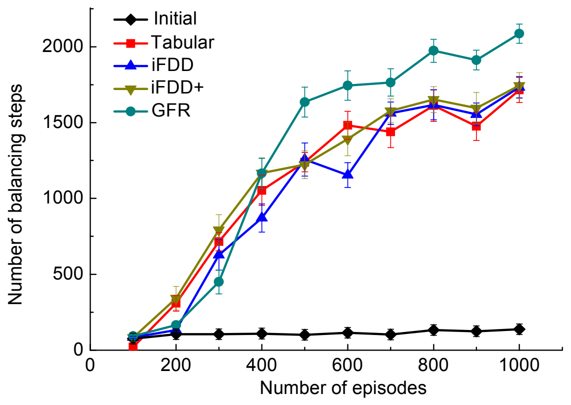

Fig. 3 The experimental results of SARSA using different representational schemes on the inverted pendulum domain, averaged over 30 independent trials

4.2 Persistent surveillance

The persistent surveillance domain shown in Fig. 4 was employed by Geramifard *et al.* (2011), which has a larger state space (approximately 150 million state–action pairs) than the inverted pendulum domain. To accomplish the surveillance task, three fuel-limited unmanned aerial vehicles (UAVs) must cooperate with each other. Each UAV has three optional actions, which are Advance, Retreat, and Loiter. There are four locations in the domain which are used to maintain the UAVs, refuel, communicate, and monitor the targets, respectively. The final state transition depends on the joint action of the three UAVs. Each UAV has initially 10 units of fuel. Every action costs a UAV a unit of fuel except when staying in the maintenance state or the refuel state. Taking the action Loiter in the refuel location can fill the fuel tank. Each UAV has an engine and a camera, and both the engine and the camera have a 5% chance to be broken at each time step. Taking the action Loiter in the maintenance area can repair the broken parts. If the engine is broken, an immediate repair is required, and the UAV has to stop all other actions to fly to the maintenance area. The camera failure does not require an immediate repair, but in this case the UAV cannot monitor the targets. The state of each UAV is a four-dimensional vector of location, fuel, engine status, and camera status. The whole state space which has 12 dimensions is the combination of the state spaces of the three UAVs.

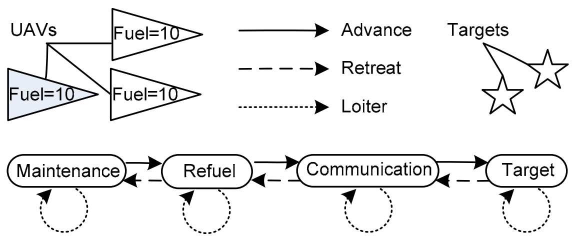

Fig. 4 Persistent surveillance (taken from Geramifard *et al.* (2011))

If one UAV is in the target state and another UAV in the communication state, the team will earn a +20 reward during every time step. If there are two UAVs in the target state with another in the communication state, the reward will be +40. If the UAVs run out of fuel outside the refuel location, the episode ends and the team receives a −50 punishment.

This time we set α_0 to 1 and found the best threshold p=2000 empirically from {1000, 2000, 3000, 4000, 5000}, keeping all the remaining parameters the same as in the previous experiment. The learner learned 10 000 episodes for each run. Fig. 5 plots the average total rewards in one episode (averaged over 30 independent runs) with the standard error of the mean. The results show that the tabular representation ran poorly, as the lack of generalization became the main obstacle to the effective application in large-scale problems. Without any expansion, the initial representation gave the worst performance as expected. Using an incremental expansion technique, GFR, iFDD, and iFDD+ showed significantly better performances than the other two methods. Compared to iFDD and iFDD+, GFR was more prominent, suggesting that the mechanism of feature replacement is an effective way for representational expansion.

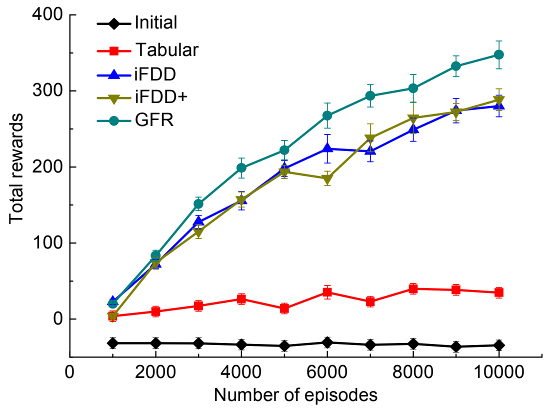

Fig. 5 The experimental results of SARSA using different representational schemes on the persistent surveillance domain, averaged over 30 independent trials

5 Conclusions and future work

GFR is a novel online representational expansion technique which can be combined with any online linear value function approximation method using binary features. The proposed algorithm expands the feature representation incrementally by discovering the replacing areas to perform feature replacement. GFR can be applied to high dimensional domains, thus overcoming the limitations present in traditional online expansion techniques. Furthermore, by identifying replacing areas, GFR can optimize the sequence in which the feature conjunctions are discovered and therefore improve the performance of representational expansion. We proved that GFR combined with TD can lead to the optimal approximation results asymptotically. We also provided the computational complexity analysis for our algorithm. In addition, the experiments of inverted pendulum and persistent surveillance empirically showed the superiority of GFR.

In future work, there is potential to employ the methods used to reduce the standard deviation and improve the stability of the algorithm. Although GFR can adapt well to large spaces and has an outstanding average performance, like other RL methods, the differences between each run are still significant. Ensuring the stability of this approach is important for the practical application of GFR.

Another area for future refinement is to expand the GFR application to real-world problems. Our success in virtual RL tasks shows that GFR has the ability to solve large space problems. However, applying our method to real problems is still a great challenge. For instance, the incomplete perception problem can cause even a fully expanded representation to be incapable of obtaining good approximation results. In this case, combining GFR with the partially observable RL approach may provide a solution to this problem and would be worth investigating.

References

Albus, J.S., 1971. A theory of cerebellar function. *Math. Biosci.*, **10**(1-2):25-61. [doi:10.1016/0025-5564(71)900 51-4]

Barto, A.G., Bradtke, S.J., Singh, S.P., 1995. Learning to act using real-time dynamic programming. *Artif. Intell.*, **72**(1-2):81-138. [doi:10.1016/0004-3702(94)00011-O]

Buro, M., 1999. From simple features to sophisticated evaluation functions. Proc. 1st Int. Conf. on Computers and Games, p.126-145. [doi:10.1007/3-540-48957-6_8]

de Hauwere, Y.M., Vrancx, P., Nowé, A., 2010. Generalized learning automata for multi-agent reinforcement learning. *AI Commun.*, **23**(4):311-324. [doi:10.3233/AIC-2010-0476]

Geramifard, A., Doshi, F., Redding, J., *et al.*, 2011. Online discovery of feature dependencies. Proc. 28th Int. Conf. on Machine Learning, p.881-888.

Geramifard, A., Dann, C., How, J.P., 2013. Off-policy learning combined with automatic feature expansion for solving large MDPs. Proc. 1st Multidisciplinary Conf. on Reinforcement Learning and Decision Making, p.29-33.

Kaelbling, L.P., Littman, M.L., Moore, A.W., 1996. Reinforcement learning: a survey. *J. Artif. Intell. Res.*, **4**:237-285. [doi:10.1613/jair.301]

Kolter, J.Z., Ng, A.Y., 2009. Near-Bayesian exploration in polynomial time. Proc. 26th Annual Int. Conf. on Machine Learning, p.513-520. [doi:10.1145/1553374. 1553441]

Lagoudakis, M.G., Parr, R., 2003. Least-squares policy iteration. *J. Mach. Learn. Res.*, **4**(6):1107-1149.

Pazis, J., Lagoudakis, M.G., 2009. Binary action search for learning continuous-action control policies. Proc. 26th Annual Int. Conf. on Machine Learning, p.793-800. [doi:10.1145/1553374.1553476]

Puterman, M.L., 1994. Markov Decision Processes—Discrete Stochastic Dynamic Programming. John Wiley & Sons, New York, NY, p.139-161.

Ratitch, B., Precup, D., 2004. Sparse distributed memories for on-line value-based reinforcement learning. Proc. 15th European Conf. on Machine Learning, p.347-358. [doi:10. 1007/978-3-540-30115-8_33]

Rummery, G.A., Niranjan, M., 1994. On-line Q-learning Using Connectionist Systems. Technical Report No. cued/ f-infeng/tr166, Engineering Department, Cambridge University.

Singh, S., Jaakkola, T., Littman, M.L., *et al.*, 2000. Convergence results for single-step on-policy reinforcement-learning algorithms. *Mach. Learn.*, **38**(3):287-308. [doi:10.1023/A: 1007678930559]

Singh, S.P., Sutton, R.S., 1996. Reinforcement learning with replacing eligibility traces. *Mach. Learn.*, **22**(1-3):123-158. [doi:10.1023/A:1018012322525]

Singh, S.P., Yee, R.C., 1994. An upper bound on the loss from approximate optimal-value functions. *Mach. Learn.*, **16**(3):227-233. [doi:10.1007/Bf00993308]

Sprague, N., Ballard, D., 2003. Multiple-goal reinforcement learning with modular sarsa(0). Proc. 18th Int. Joint Conf. on Artificial Intelligence, p.1445-1447.

Sturtevant, N.R., White, A.M., 2006. Feature construction for reinforcement learning in hearts. Proc. 5th Int. Conf. on Computers and Games, p.122-134. [doi:10.1007/978-3-540-75538-8_11]

Sutton, R.S., 1996. Generalization in reinforcement learning: successful examples using sparse coarse coding. *Adv. Neur. Inform. Process. Syst.*, **8**:1038-1044.

Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning: an Introduction. MIT Press, Cambridge, MA, USA, p.3-25.

Tsitsiklis, J.N., 1994. Asynchronous stochastic approximation and *Q*-learning. *Mach. Learn.*, **16**(3):185-202. [doi:10. 1007/Bf00993306]

Tsitsiklis, J.N., van Roy, B., 1997. An analysis of temporal-difference learning with function approximation. *IEEE Trans. Automat. Contr.*, **42**(5):674-690. [doi:10.1109/9. 580874]

Watkins, C.J.C.H., Dayan, P., 1992. *Q*-learning. *Mach. Learn.*, **8**(3-4):279-292. [doi:10.1007/Bf00992698]

Whiteson, S., Taylor, M.E., Stone, P., 2007. Adaptive Tile Coding for Value Function Approximation. Technical Report No. AI-TR-07-339, University of Texas at Austin.