Generalized Compatible Function Approximation for Policy Gradient Search

Yiming Peng[1(✉)], Gang Chen[1], Mengjie Zhang[1], and Shaoning Pang[2]

[1] School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand
{yiming.peng,aaron.chen,mengjie.zhang}@ecs.vuw.ac.nz
[2] Department of Computing, Unitec Institute of Technology, Auckland, New Zealand
ppang@unitec.ac.nz
http://www.dmli.info, http://ecs.victoria.ac.nz/Groups/ECRG/

Abstract. Reinforcement learning aims at solving stochastic sequential decision making problems through direct trial-and-error interactions with the learning environment. In this paper, we will develop generalized compatible features to approximate value functions for reliable Reinforcement Learning. Further guided by an Actor-Critic Reinforcement Learning paradigm, we will also develop a generalized updating rule for policy gradient search in order to constantly improve learning performance. Our new updating rule has been examined on several benchmark learning problems. The experimental results on two problems will be reported specifically in this paper. Our results show that, under suitable generalization of the updating rule, the learning performance and reliability can be noticeably improved.

Keywords: Markov decision process · Reinforcement learning · Actor critic · Policy gradient search · Function approximation · Generalization · Compatible feature

1 Introduction

Stochastic sequential decision making problems frequently appear in diverse real-world applications and are increasingly gaining attentions [8]. These problems can be described through Markov Decision Processes (MDPs) [8]. To solve an MDP, traditional techniques such as dynamic programming can be applied but they often suffer from the issue of *curse of dimensionality* whenever the state space is large. In this situation, an alternative technique, known as Reinforcement Learning (RL), is widely considered to be more suitable for practical use [5].

In RL, to break the curse, a common approach is to adopt function approximation for value functions (aka. critic) via parameterization, the dimension of which is generally smaller than that of states [1,6]. Clearly the success of RL depends on appropriate and generalized function approximators [1,6,7]. Based on properly approximated value functions, we can further pursue the solutions to MDPs in the form of parametric policies (aka. actor). Under this *Actor-Critic*

A. Hirose et al. (Eds.): ICONIP 2016, Part I, LNCS 9947, pp. 615–622, 2016.
DOI: 10.1007/978-3-319-46687-3_68

(AC) framework, various policy gradient search methods can be implemented [1,5]. In particular, through iterative updates to *policy parameters*, the learning performance is expected to be constantly improved [5].

Guided by the AC framework, Sutton proved the first time that the gradient of J with respect to policy parameters $\boldsymbol{\theta}$ follows the formula below [6],

$$\Delta\boldsymbol{\theta} \propto \nabla_{\boldsymbol{\theta}} J^{\pi}(\boldsymbol{\theta}) = \sum d^{\pi}(\boldsymbol{s}) \sum \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{s}, a) Q^{\pi}(\boldsymbol{s}, a), \tag{1}$$

where J^{π} (see (4)) stands for the *expected long term payoff* obtainable by an agent upon following policy π which is parameterized by $\boldsymbol{\theta}$ (i.e. the policy parameters). In (1), $\boldsymbol{s}$ and a represent the state and action of an MDP respectively (for more information on MDP, please refer to Sect. 2). $Q^{\pi}(\boldsymbol{s}, a)$ is the expected total reward while initially taking action a at state $\boldsymbol{s}$ following policy π. To use (1) for policy gradient search, a key issue is to approximate $Q^{\pi}(\boldsymbol{s}, a)$. In [6], Sutton proposed an important method to estimate $Q^{\pi}(\boldsymbol{s}, a)$, i.e.

$$Q^{\pi}(\boldsymbol{s}, a) \approx \hat{Q}^{\pi}(\boldsymbol{s}, a) = \boldsymbol{\omega}^{T} \Phi(\boldsymbol{s}, a) = \boldsymbol{\omega}^{T} \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{s}, a), \tag{2}$$

where $\boldsymbol{\omega}$ is used to parameterize $\hat{Q}^{\pi}$, and $\Phi(\boldsymbol{s}, a) = \nabla_{\boldsymbol{\theta}} \ln \pi(\boldsymbol{s}, a)$ is the so-called *compatible feature vector*. Furthermore, Sutton proved that the approximation of Q^{π} by $\hat{Q}^{\pi}$ in (2) will still ensure precise evaluation of $\nabla_{\boldsymbol{\theta}} J^{\pi}$ in (1), i.e.

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &\equiv \sum d^{\pi}(\boldsymbol{s}) \sum \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{s}, a) \hat{Q}^{\pi}(\boldsymbol{s}, a) \\ &= \sum d^{\pi}(\boldsymbol{s}) \sum \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{s}, a) \boldsymbol{\omega}^{T} \Phi(\boldsymbol{s}, a) \end{aligned} \tag{3}$$

Based on (3), many policy gradient search algorithms follow strictly (2) to approximate Q^{π} [1]. However, we found that, under proper conditions (to be detailed in a separate venue due to space limitation), when the compatible feature vector in (2) is represented as $\Phi(\boldsymbol{s}, a) = \nabla_{\boldsymbol{\theta}} \mathcal{G}(\pi(\boldsymbol{s}, a), \nu)$ where $\mathcal{G}(\cdot)$ is a generalization of the logarithm function and ν controls the level of generalization, the result in (3) will continue to hold. In the literature, $\mathcal{G}(\cdot)$ has been utilized to define *Tsallis entropy* with substantial practical applications in many disciplines [2]. Inspired by this understanding, we seek to take the first step towards answering an important research question: *when the generalized logarithm function $\mathcal{G}(\cdot)$ is used to produce the compatible feature Φ in (2), will policy gradient search algorithm become more effective and reliable for RL?* To the utmost of our knowledge, so far this research question hasn't been studied sufficiently.

In this paper, based on an incremental regular-gradient actor-critic algorithm (i.e. RAC) proposed by Bhatnagar et al. in [1], we will investigate experimentally the usefulness of $\mathcal{G}(\cdot)$ for building compatible features Φ and develop some empirical answer to our research question. Several benchmark problems, including Puddle World [5] and Cart-Pole [5], will be employed for this purpose. Our experiment results clearly show that, under suitable generalization, the learning performance can exhibit observable improvement.

2 Markov Decision Process

An MDP is described as an agent repetitively interacting with a stochastic environment at discrete time intervals [5]. At any time t, the agent can observe its

state $\boldsymbol{s}_t \in \mathcal{S}$ in the environment and choose an action $a_t \in \mathcal{A}$ to execute, where $\mathcal{S}$ and $\mathcal{A}$ (i.e., state and action spaces) represent countable sets of all possible states and actions, respectively. Meanwhile, the environment responds with an arbitrary scalar reward $r(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1})$, depending partially on the next state $\boldsymbol{s}_{t+1}$ governed by the state transition probability $P(\boldsymbol{s}_t, a_t, \boldsymbol{s}_t)$ [5]. The learning objective in an MDP is to achieve the maximum *long term payoff*, which can be modeled in several different ways [5]. In this paper, we focus specifically on the discounted infinite horizon model. Accordingly, the *expected long term payoff* of policy π can be formulated as,

$$J^\pi = V^\pi(\boldsymbol{s}_0) = \mathbf{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | \boldsymbol{s}_t = \boldsymbol{s}_0]. \tag{4}$$

where $\gamma \in [0, 1)$ is the *discount factor*. Meanwhile r_{t+k+1} is the immediate reward at time step $k+1$ provided that the agent started from state $\boldsymbol{s}_t$. Similarly, function Q^π introduced in Sect. 1 can be defined as below,

$$Q^\pi(\boldsymbol{s}, a) = \mathbf{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | \boldsymbol{s}_t = \boldsymbol{s}, a_t = a]. \tag{5}$$

Based on (4) and (5), we can re-formulate the expected long term payoff starting from any state $\boldsymbol{s}$ as

$$V^\pi(\boldsymbol{s}) = \sum_{a \in \mathcal{A}} \pi(\boldsymbol{s}, a) Q^\pi(\boldsymbol{s}, a). \tag{6}$$

Hence, the ultimate goal of RL is to identify the optimal policy π^* below,

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, V^\pi(\boldsymbol{s}_0). \tag{7}$$

3 A Regular-Gradient Actor-Critic Algorithm

In this work, we decide to base our experimental study on an incremental regular-gradient actor-critic algorithm (i.e. RAC) proposed in [1]. We choose RAC due to its simplicity and proven effectiveness on many benchmark RL problems. According to [1,6], Q^π is approximated by $\hat{Q}_\pi$ in RAC by minimizing the *Mean Square Error* (MSE),

$$\epsilon^\pi(\boldsymbol{\omega}) = \sum_{\boldsymbol{s} \in \mathcal{S}} d^\pi(\boldsymbol{s}) \sum_{a \in \mathcal{A}} \pi(\boldsymbol{s}, a)[Q^\pi(\boldsymbol{s}, a) - \hat{Q}^\pi(\boldsymbol{s}, a)]^2. \tag{8}$$

The minimization can be achieved in theory by solving the equation below

$$\nabla_{\boldsymbol{\omega}} \epsilon^\pi(\boldsymbol{\omega}) = \sum_{\boldsymbol{s} \in \mathcal{S}} d^\pi(\boldsymbol{s}) \sum_{a \in \mathcal{A}} \pi(\boldsymbol{s}, a)[Q^\pi(\boldsymbol{s}, a) - \hat{Q}(\boldsymbol{s}, a)] \nabla_{\boldsymbol{\omega}} \hat{Q}^\pi(\boldsymbol{s}, a) = 0, \tag{9}$$

Because of (2), $\nabla_{\boldsymbol{\omega}} \hat{Q}^\pi(\boldsymbol{s}, a) = \Phi(\boldsymbol{s}, a)$ in (9). Following (9), the temporal difference (TD) error δ is

$$\delta_t^\pi = r(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}) + \gamma^{t+1} V^\pi(\boldsymbol{s}_{t+1}) - V^\pi(\boldsymbol{s}_t) \tag{10}$$

By defining ${\boldsymbol{v}^\pi}^T\phi(\boldsymbol{s})$ as an unbiased estimation of the value function $V^\pi(\boldsymbol{s})$, where $\boldsymbol{v}^\pi$ is the so-called *value-function parameters* and $\phi(\boldsymbol{s})$ is another set of *state features* to be distinguished from $\Phi(\boldsymbol{s}, a)$, the TD error can be further described as,

$$\delta_t^\pi = r(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}) + \gamma^{t+1}{\boldsymbol{v}_{t+1}^\pi}^T\phi(\boldsymbol{s}_{t+1}) - {\boldsymbol{v}_t^\pi}^T\phi(\boldsymbol{s}_t) \tag{11}$$

Consequently, following (11) and (4), we can eventually determine the formula in (12) for incremental update of $\boldsymbol{v}^\pi$.

$$\boldsymbol{v}_{t+1}^\pi \leftarrow \boldsymbol{v}_t^\pi + \alpha \cdot \delta_t^\pi \cdot \phi(\boldsymbol{s}_t), \tag{12}$$

where α is a learning rate. Subsequently, the updating formula in (13) for incrementally updating the policy parameters $\boldsymbol{\theta}$ can also be determined easily.

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \beta \cdot \delta_t^\pi \cdot \Phi(\boldsymbol{s}_t, a_t), \tag{13}$$

Similar to (12), β in (13) refers to a separate learning rate. To summarize our discussion above, Algorithm 1 presents the pseudo-code for RAC. It is to be noted that, for RAC, the *compatible feature* is computed from $\Phi(\boldsymbol{s}_t, a_t) = \frac{1}{\pi(\boldsymbol{s},a)}\nabla_{\boldsymbol{\theta}}\pi(\boldsymbol{s}, a)$. In the next section, we will generalize the calculation of $\Phi(\boldsymbol{s}_t, a_t)$ by using the generalized logarithm function $\mathcal{G}(\cdot)$.

Algorithm 1. Regular-Gradient Actor-Critic Algorithm [1]

Input: an MDP $\langle \mathcal{S}, \mathcal{A}, P(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}), r(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}), \gamma \rangle$
Output: $\boldsymbol{\theta}$, $\boldsymbol{v}^\pi$, $\phi(\boldsymbol{s})$

Initialization:
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$
$\boldsymbol{v}^\pi \leftarrow \boldsymbol{v}_0^\pi$
$\boldsymbol{s} \leftarrow \boldsymbol{s}_0$
$a \leftarrow a_0$
Learning Process:
for $t = 0, 1, 2, \ldots$ **do**
  $\delta_t^\pi \leftarrow r(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}) + \gamma{\boldsymbol{v}_{t+1}^\pi}^T\phi(\boldsymbol{s}_{t+1}) - {\boldsymbol{v}_t^\pi}^T\phi(\boldsymbol{s}_t)$
  $\boldsymbol{v}_{t+1}^\pi \leftarrow \boldsymbol{v}_t^\pi + \alpha\delta_t^\pi\phi(\boldsymbol{s}_t)$
  $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \beta\delta_t^\pi\Phi(\boldsymbol{s}_t, a_t)$
end for
return $\boldsymbol{\theta}$, $\boldsymbol{v}^\pi$

4 Generalized Compatible Function Approximation

In this research, we focus primarily on generalizing the computation of the *compatible feature* $\Phi(\boldsymbol{s}, a)$. The learning procedure still follows closely Algorithm 1.

As we explained in Sect. 1, we propose to use a generalized logarithm function $\mathcal{G}(\cdot)$ as shown below to determine the compatible feature Φ in (2),

$$\mathcal{G}(\pi(\boldsymbol{s},a),\nu) = \frac{\pi(\boldsymbol{s},a)^{1-\nu}-1}{1-\nu}. \tag{14}$$

Clearly, the level of generalization in $\mathcal{G}(\cdot)$ is controlled by ν. Whenever $\nu = 1$, $\mathcal{G}(\cdot)$ degrades to the standard logarithm function, i.e.

$$\lim_{\nu\to 1}\mathcal{G}(\pi(\boldsymbol{s},a),\nu) = \ln\pi(\boldsymbol{s},a)\,. \tag{15}$$

For simplicity, ν in (14) will be called the *compatible generalization factor*. In consequence, we can obtain a new way to determine the compatible feature below,

$$\begin{aligned}\tilde{\Phi}(\boldsymbol{s},a) &= \nabla_{\boldsymbol{\theta}}\mathcal{G}(\pi(\boldsymbol{s},a),\nu)\\ &= \nabla_{\boldsymbol{\theta}}\pi(\boldsymbol{s},a)\pi(\boldsymbol{s},a)^{1-\nu},\end{aligned} \tag{16}$$

Given a Gaussion Distribution for policy $\pi(\boldsymbol{s},a)$, we have,

$$\pi(\boldsymbol{s},a) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(a-\boldsymbol{\mu})^2}{2\sigma^2}}, \tag{17}$$

where $\boldsymbol{\mu} = \boldsymbol{\theta}^T\phi(s)$ is the mean action output from policy π in state $\boldsymbol{s}$, which can be adjusted by changing policy parameters $\boldsymbol{\theta}$. On the other hand, the standard deviation σ in (17) is pre-determined. Note that π in RHS of (17) refers to the circumference ratio.

Based on the generalized compatible feature in (16), the updating rule shown in (13) is now modified to become

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &\leftarrow \boldsymbol{\theta}_t + \beta\cdot\eta\cdot\delta_t^{\pi}\cdot\tilde{\Phi}(\boldsymbol{s}_t,a_t)\\ &= \boldsymbol{\theta}_t + \beta\cdot\eta\cdot\delta_t^{\pi}\nabla_{\boldsymbol{\theta}}\pi(\boldsymbol{s}_t,a_t)\pi(\boldsymbol{s}_t,a_t)^{1-\nu}\,,\end{aligned} \tag{18}$$

where a new constant factor η is introduced in (18) to ensure that $\boldsymbol{\theta}$ will be updated at the same scale in Algorithm 1 regardless of using either updating rule (13) or (18). For this purpose, we need to solve the following equation,

$$\mathbf{E}_a[|\nabla_{\boldsymbol{\theta}}\ln\pi(\boldsymbol{s},a)|] = \mathbf{E}_a[|\eta\nabla_{\boldsymbol{\theta}}\mathcal{G}(\pi(\boldsymbol{s},a),\nu))|]. \tag{19}$$

From (19), we can directly determine the value for η below,

$$\eta = \begin{cases} \dfrac{(2\pi)^{\frac{1-\nu}{2}}(\nu-2)\sigma^{1-\nu}\left(1-e^{-\frac{(\mu+t)^2}{2\sigma^2}}\right)}{e^{\frac{(\nu-2)(t-\mu)^2}{2\sigma^2}}+e^{\frac{(\nu-2)(\mu+t)^2}{2\sigma^2}}-2}, & 0<\nu<2\\ \dfrac{\sqrt{\frac{2}{\pi}}\sigma\left(1-e^{-\frac{(\mu+t)^2}{2\sigma^2}}\right)}{(\mu+t)^2}, & \nu=2 \end{cases}. \tag{20}$$

Again π in (20) is the circumference ratio. Meanwhile t and $-t$ give the upper and lower bounds for the action output from any policy, respectively.

Table 1. Experiment general settings for one trial.

Problem	Training		Testing		Evaluating ν values
	Episodes	Steps	Episodes	Steps	
Puddle world	10000	1000	50	100	{0.5, 0.7, 0.9, **1.0**, 1.1, 1.5, 2.0}
Cart pole	20000	50	50	1000	{0.5, 0.7, 0.9, **1.0**, 1.1, 1.5, 2.0}

5 Experimental Results

To understand the efficacy of using generalized updating rule for learning policy parameters in (18), we will evaluate the performance and reliability of RAC on two benchmark problems. In order to obtain reliable results, 50 independent trials will be performed on each benchmark problem and with respect to different settings of the compatible generalization factor ν (note that when $\nu = 1$, the original updating rule in (13) is realized). A total of 8 different settings for ν have been examined in our experiments (see Table 1). To simplify our discussion, in this section, CASE-X will denote the experiments on RAC when $\nu = X$.

Common settings that we follow on every trial have been summarized in Table 1. As evidenced in this table, after every 20 training episodes during each trial, the learned policy will be further tested on 50 independent testing episodes to measure learning performance.

5.1 Experiments on the Puddle World Problem

To compare the performance differences among various settings of ν, we firstly evaluate RAC on the classic Puddle World problem [3,5]. Figure 1 presents the average steps to reach the goal region upon using the policies learned through RAC. As seen from Fig. 1, near-optimal policies can be learned successfully whenever $\nu \in [0.9, 2.0]$. On the other hand, CASE-0.5 and CASE-0.7 failed to solve this problem satisfactorily. Among all the results presented in Fig. 1, CASE-1.5 appears to achieve the best performance by observation (i.e. on average 19.2 steps to reach the goal region). In comparison, for CASE-1.0 (i.e. normal RAC), the average steps to reach the goal region is 46.4 after 10,000 learning episodes have been completed. A t-test is performed in between CASE-1.0 and CASE-1.5 and it produces a p-value of 0.10, insufficient to prove that CASE-1.5 is significantly better. However, we found that the problem is solved successfully by CASE-1.5 100 % of the time. For CASE-1.0, the problem is solved on only 94 % of the trials. This observation suggests that CASE-1.5 can solve the problem more reliably.

5.2 Experiments on the Cart-Pole Problem

Next, we have compared the learning performances on the Cart-Pole problem (aka. inverted pendulum problem) [5]. The performances in terms of the average

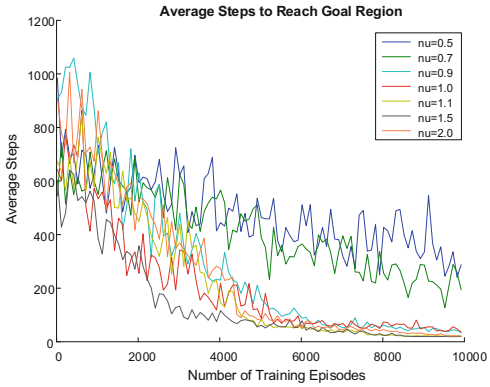
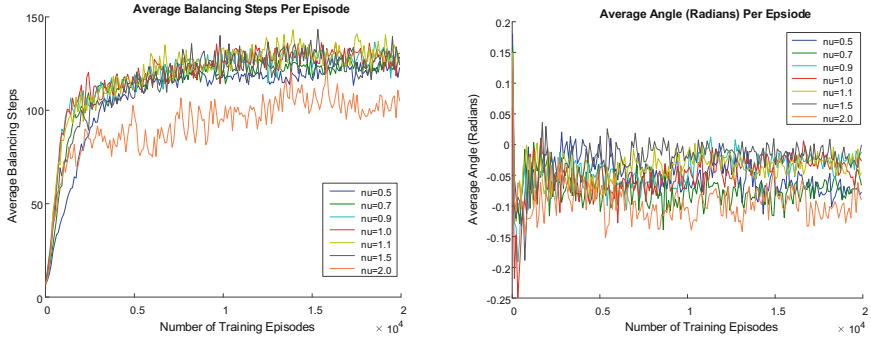


Fig. 1. Average steps to goal region.

ξ (i.e. the angle of the pole) and the average balancing steps (i.e. the duration for the pole to be balanced continually) are presented in Fig. 2. It can be observed in Fig. 2(a) that, in comparison to other cases, during a long learning period from 3000 training episodes to the end, CASE-1.5 can manage to bring the pole closer to the upright position on average. For example, at 3000 training episodes, the average ξ achieved by CASE-1.5 is -0.01. In comparison, CASE-1.0 can only manage to achieve on average of -0.07 for ξ. However, this observed performance difference is not verifiable through statistical tests (perhaps more repeated tests are to be performed in order to reveal significant differences in between CASE-1.0 and CASE-1.5).

On the other hand, by checking the average balancing steps in Fig. 2(b), we found that most of the cases can solve this problem reasonably well. The only



(a) (b)

Fig. 2. The performance evaluation for the Cart-Pole problem. Part (a) shows the average ξ, and part (b) shows the average balancing steps.

case that falls apart is when $\nu = 2.0$, suggesting that the value for ν cannot significantly differ from 1.0.

5.3 Experiment Summary

In summary, our experimental evaluation clearly shows that there exists a strong correlation in between the degree of generalization in (18) and the learning performance as well as reliability. Whenever we set ν in (18) to a proper value, e.g. 1.5, observable improvement on learning performance and reliably can be witnessed. However, when ν deviates significantly from the common setting of 1.0, the learning performance may degrade abruptly.

6 Conclusions

In this paper we studied the possibility of using generalized compatible features for value function approximation. Guided by an Actor-Critic framework for RL, a generalized rule for policy gradient search in RL algorithms has been developed successfully. We have experimentally examined the usefulness of this rule on two benchmark RL problems. We found that, under suitable generalization (e.g. $\nu = 1.5$), the learning performance and reliability can be noticeably improved. We have therefore obtained some promising and empirical answer to the research question highlighted in Sect. 1. Based on the research results reported in this paper, we will further explore our research question in the future through in-depth theoretical analysis and extensive experimental assessments.

Acknowledgments. Authors appreciate all the supports from NeSI [4], who provides the High Performance Computing facility to ensure the success of our computationally heavy experiments.

References

1. Bhatnagar, S., Sutton, R.S., Ghavamzadeh, M., Lee, M.: Natural actor-critic algorithms. Automatica **45**(11), 2471–2482 (2009)
2. Cartwright, J.: Roll over, boltzmann. Phys. World **27**(5), 31–35 (2014)
3. Chen, G., Douch, C.I.J., Zhang, M.: Reinforcement learning in continuous spaces by using learning fuzzy classifier systems. IEEE Trans. Evol. Comput. **PP**(99), 1 (2016)
4. NeSI: New Zealand eScience Infrastructure (2016). https://www.nesi.org.nz/
5. Sutton, R.S., Barto, A.G.: Reinforcement Learning : An Introduction. MIT Press, Cambridge (1998)
6. Sutton, R.S., Mcallester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. Adv. Neural Inf. Process. Syst. **12**, 1057–1063 (1999)
7. Sutton, R.: Generalization in reinforcement learning: successful examples using sparse coarse coding. In: Advances in Neural Information Processing Systems, pp. 1038–1044 (1996)
8. White, D.J.: A survey of applications of Markov decision processes. J. Oper. Res. Soc. **44**, 1073–1096 (1993)