ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence
and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Application of Newton's Method to Action Selection in Continuous State- and Action-Space Reinforcement Learning

Barry D. Nichols and Dimitris C. Dracopoulos

School of Science and Technology
University of Westminster, 115 New Cavendish St, London, W1W 6XH - England

Abstract. An algorithm based on Newton's Method is proposed for action selection in continuous state- and action-space reinforcement learning without a policy network or discretization. The proposed method is validated on two benchmark problems: Cart-Pole and double Cart-Pole on which the proposed method achieves comparable or improved performance with less parameters to tune and in less training episodes than CACLA, which has previously been shown to outperform many other continuous state- and action-space reinforcement learning algorithms.

1 Introduction

When reinforcement learning (RL) [1] is applied to problems with continuous state- and action-space it is impossible to compare the values of all possible actions; thus selecting the greedy action a from the set of all actions $\mathcal{A}$ available from the currents state s requires solving the optimization problem:

$$\arg\max_a Q(s,a), \quad \forall a \in \mathcal{A} \tag{1}$$

Although it has been stated that if $\nabla_a Q(s,a)$ is available it would be possible to apply a gradient based method [2], it is often asserted that directly solving the optization problem would be prohibitively time-consuming [3, 4]. Here we investigate the possibility of applying Newton's Method to solving (1), and present results obtained on two benchmark problems from the literature: Cart-Pole and double Cart-Pole. Performance is compared to a state-of-the-art method: continuous actor critic learning automaton (CACLA) which has previously been shown to outperform other techniques on the Cart-Pole problem [5].

2 Current Methods

There are several approaches to applying RL to problems with continuous state- and action-space, and although they all enable the application of RL to continuous problems they also all have some drawbacks:

Action-space discretization [2] the granularity of which must be determined to allow sufficient representation of the value function whilst ensuring $|\mathcal{A}|$ remains small enough to allow evaluation of all actions; this is addressed, to some extent, by [6] which only selects from two actions $a \pm a_{\Delta}$, however, this significantly limits the possible actions; interpolation [7] suffers from a similar trade off between sufficient wires to represent the value function without requiring too much computation; actor-critic methods [4] require two function approximators, which must both be trained; direct policy search [8, 9] requires many episodes, which may be expensive, in order to optimize the policy function without taking advantage of rewards received at each time-step.

3 Proposed Method

The method proposed here applies the well known iterative optimization technique Newton's Method (NM) [10] to solving (1) in order to select the greedy action a, by taking advantage of the ability to quickly calculate $\nabla_a Q(s,a)$ and $\nabla_a^2 Q(s,a)$ when a multilayer perceptron (MLP) is applied to the approximation of the value function. By setting s to the current state, we are able to solve (1) using NM, as shown in Algorithm 1.

In the following experiments a single hidden layer MLP was used with 12 nodes in the hidden layer, with a tanh activation function and a linear output layer. The SARSA algorithm [1] was applied to calculate the error used to update the value function, and backpropagation (with momentum of 0.75) was utilized in updating the weights; hence, we shall refer to the proposed method as NM-SARSA.

Algorithm 1 Newton's Method Action Selection

```
procedure GetActionNM($s$)
    $a_{best} \leftarrow 0$
    for all $a_0 \in \{a_{min}, a_{min} + a_\Delta, \dots, a_{max} - a_\Delta, a_{max}\}$ do
        $a \leftarrow a_0$, $a_{previous} \leftarrow a_{max} + 10$
        for $i \leftarrow 1, max\_iterations$ do
            $a \leftarrow \left[a - \frac{\partial Q(s,a)/\partial a}{\partial^2 Q(s,a)/\partial a^2}\right]_{a_{min}}^{a_{max}}$      ▷ Limit $a$ to allowable range
            if $Q(s,a) > Q(s,a_{best})$ then
                $a_{best} \leftarrow a$
            end if
            if $|a - a_{previous}| < 0.001$ then
                break                                   ▷ If $a$ has converged move on to next $a_0$
            end if
            $a_{previous} \leftarrow a$
        end for
    end for
    return $a_{best}$
end procedure
```

As few iterations of NM were required to converge, this algorithm does not introduce a significantly large run-time overhead compared with discretized methods, whilst allowing any $a \in [a_{min}, a_{max}]$. As NM-SARSA is not limited to the discrete actions $a_{min} + n \cdot a_\Delta$, which are merely the starting points for NM, it is possible to apply a far larger a_Δ for this algorithm than with action-space discretization methods (here $a_\Delta = 0.5$ was applied to selecting $a \in [-1, 1]$).

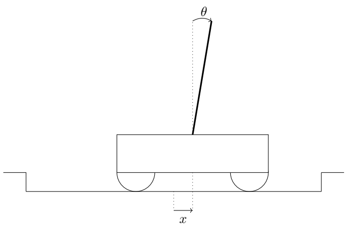
A maximum of 10 iterations of NM was used; however, if there was no significant change in a the NM loop was terminated immediately. Also, as a sometimes reached a point where it was switching between 2 or more values, a_{best} was updated at every iteration to ensure the maximum action was returned.

4 Experiments

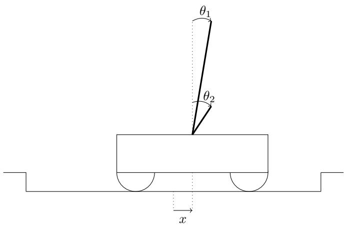
The Cart-Pole problem and double pole variant were used to evaluate the performance of NM-SARSA. In both problems the aim is for the agent to learn a control policy to balance the pole(s) for a specified duration of time whilst remaining within the boundaries of the track.

Both experiments were conducted without noise, with Gaussian noise and also with uniform noise. Noise was added to the action after it was limited to the allowable range; thus the applied force may exceed that which was available to the agent. The agent was not made aware of noise, and therefore would update the value and policy functions based on the selected action rather than the action applied to the environment. Gaussian noise was generated from the normal distribution and multiplied by F_{max}, whereas uniform noise was randomly selected from $[-F_{max}/2, F_{max}/2]$.

For each episode a simulation was run for a maximum of 120 simulated seconds and was terminated immediately if either the pole (one of the poles) fell or the cart reached the edge of the track. The pole was considered to have fallen if $|\theta| > \pi/15$, and the cart reached the edge of the track if $|x| > 2.4$. Every 0.02 simulated seconds the RL agent selected an action; noise was added to the action; the environment was updated using the Runge Kutta fourth order method and the reward was calculated. The reward was -1 if the pole fell or the cart reached the edge of the track and 0 otherwise.



(a) Diagram of the Cart-Pole problem.



(b) Diagram of the double Cart-Pole problem.

The parameters of the RL agents for both experiments are listed in Table 1. These parameters were found to perform best of all those experimented with on the noise-free setting, but were not re-tuned when noise was added.

Parameter	Cart-Pole		Double Cart-Pole	
	NM-SARSA	CACLA	NM-SARSA	CACLA
Learn rate	0.3	0.1	0.2	0.1
RL step-size (α)	0.2	0.1	0.2	0.2
CACLA variance (β)	N/A	0.001	N/A	0.001
RL discount rate (γ)	0.9	0.8	0.9	0.9
Exploration	Gaussian	Gaussian	none	Gaussian

Table 1: RL Agent Parameters

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7. Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

4.1 Cart-Pole Balance Problem

The Cart-Pole problem is a widely used control benchmark problem [5, 11], but often the actions are limited to $\mathcal{A} = \{0, \pm 10\}$N; here, however, continuous actions are permitted. The equations of motion used to update the environment were as [11].

The state vector comprised the pole angle; pole angular velocity; cart distance from centre of track; and cart velocity $s = [\theta, \dot{\theta}, x, \dot{x}]^\top$. The action was the force applied to the cart $a = F \in [-10, 10]$N. The initial state for each episode was $[0 + o, 0, 0, 0]^\top$, where o was uniformly generated offset in the range $[-0.05, 0.05]$. The exploration schedule used for both CACLA and NM-SARSA was implemented by multiplying the random exploration value by a coefficient before applying it to the action. This exploration coefficient was set to 1 at the start of each episode and was reduced at each time-step by 0.001 until it reached 0.

4.2 Double Cart-Pole Balance Problem

The double Cart-Pole problem is an extension of the standard Cart-Pole problem, whereby the cart has two poles, of differing lengths, both of which must be balanced. The simulation in this experiment was the same as that of [4] except the maximum time of the simulation was 120s (rather than 20s) making the task more challenging.

The state vector comprised the angle and angular velocity of each pole; cart distance from centre of track; and cart velocity $s = [\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2, x, \dot{x}]^\top$, and the action was the force applied to the cart $a = F \in [-40, 40]$N. The initial state for each episode was $[\frac{\pi}{180}, 0, 0, 0, 0, 0]^\top$ (as [4]).

The exploration magnitude used by CACLA was kept constant for all episodes, but every tenth episode was run without exploration to determine when learning has been achieved. This was found to perform better than any exploration schedules tested. The exploration size was smaller than was used in [4] as in those experiments it was orders of magnitude larger than the maximum action thereby forcing the agent to learn a bang-bang control policy.

5 Results

Results were produced from 50 different runs of NM-SARSA and CACLA. As soon as the agent succeeded in one of the episodes no further training took place; however, 10 testing episodes were carried out with no training or exploration in order to test the performance of the controller. The percentage of runs in which the agent succeeded for all 10 testing episodes is the testing success rate.

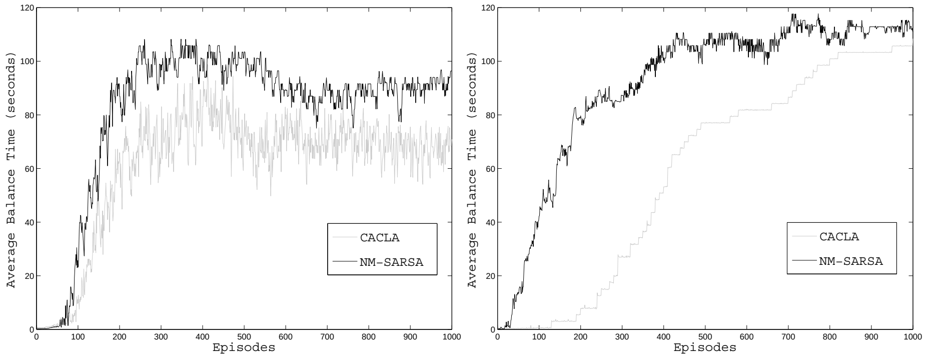
Table 2 shows the success rate, median, minimum and maximum number of episodes taken by each method to successfully balance the Cart-Pole for 120s, and also the testing success rate. There were no testing episodes when noise was not applied to the double Cart-Pole as the initial state was the same for every episode.

Problem	Method	Noise	Success	Episodes to Train			Testing
			Rate	Median	Min	Max	Success Rate
Single Pole	CACLA	None	100%	129	68	267	98%
		Gaussian	100%	136	76	314	92%
		Uniform	100%	149	84	269	90%
	NM-SARSA	None	100%	98	52	215	98%
		Gaussian	100%	79	39	127	94%
		Uniform	100%	81	48	155	90%
Double Pole	CACLA	None	96%	430	20	970	N/A
		Gaussian	94%	430	180	970	90%
		Uniform	40%	720	290	980	32%
	NM-SARSA	None	100%	89	2	464	N/A
		Gaussian	100%	106	50	639	72%
		Uniform	98%	131	49	602	72%

Table 2: Results

As can be seen from Table 2, NM-SARSA achieves comparable performance with CACLA, but consistently trains in less episodes. On the double pole problem there was an even greater difference in the number of episodes required to train NM-SARSA compared to CACLA, also the success rate of NM-SARSA was a significant improvement over that of CACLA when uniform noise was used.

The plots in Figure 2 were produced on the noise-free simulation, where 1000 episodes were run regardless of whether the agent could balance the pole(s) in order to calculate the average balance time per episode averaged over the 50 runs. In the Cart-Pole problem (Figure 2a), although training is similar, the average balance time is always slightly longer for NM-SARSA. As no exploration reduction was used for CACLA on the double Cart-Pole problem exploration was set to zero once it was able to balance for all subsequent episodes to avoid the average times being close to zero due to exploratory actions. It can be seen from both Table 2 and Figure 2b that NM-SARSA trains in considerably fewer episodes than CACLA on this problem.



(a) Cart-Pole task.

(b) Double Cart-Pole task.

Fig. 2: Balance time per episode, averaged over 50 runs.

6 Conclusion

A simple method of action selection for continuous state- and action-space RL was proposed which utilizes NM to take advantage of the first and second partial derivatives of $Q(s,a)$ w.r.t. a. The performance of the proposed method was tested empirically on two difficult continuous-action control benchmark problems from the literature, where it was shown to train in less episodes than a state-of-the-art method with less parameters to tune and is able to update without exploratory actions, reducing the requirement of tuning the exploration schedule. NM-SARSA also performed well with the addition of noise applied to the selected action, where it even exceeded the performance of CACLA on the double Cart-Pole problem. This came at the cost of increased action selection time, which was still less than 0.002s in these experiments.

Future work seeks to determine the applicability of derivative-free optimisation techniques, e.g. evolutionary algorithms, to solving (1) which would be applicable even when the derivatives of $Q(s,a)$ are unavailable.

References

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, Massachusetts, 1998.

[2] Juan C. Santamarí, Richard S. Sutton, and Ashwin Ram. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive behavior*, 6(2):163–217, 1997.

[3] I. Grondman, L. Busoniu, G. A D Lopes, and R. Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(6):1291–1307, 2012.

[4] Hado van Hasselt. Reinforcement learning in continuous state and action spaces. In Marco Wiering and Martijn Otterlo, editors, *Reinforcement Learning*, volume 12 of *Adaptation, Learning, and Optimization*, pages 207–251. Springer Berlin Heidelberg, 2012.

[5] Hado van Hasselt and Marco A. Wiering. Reinforcement learning in continuous action spaces. In *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on*, pages 272–279, April 2007.

[6] Jason Pazis and Michail G. Lagoudakis. Learning continuous-action control policies. In *Adaptive Dynamic Programming and Reinforcement Learning, 2009. ADPRL '09. IEEE Symposium on*, pages 169–176, April 2009.

[7] Leemon C. Baird and Harry Klopf. Reinforcement learning with high-dimensional, continuous actions. Technical report, Wright Laboratory, 1993.

[8] Dimitris Dracopoulos, Dimitrios Effraimidis, and Barry D. Nichols. Genetic programming as a solver to challenging reinforcement learning problems. volume 8 of *Horizons in Computer Science Research*, pages 145–174. Nova Publications, Hauppauge, NY, USA, 2013.

[9] M. Riedmiller, J. Peters, and S. Schaal. Evaluation of policy gradient methods and variants on the cart-pole benchmark. In *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on*, pages 254–261, April 2007.

[10] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* Springer, August 2000.

[11] Jennie Si and Yu-Tsung Wang. Online learning control by association and reinforcement. *Neural Networks, IEEE Transactions on*, 12(2):264–276, March 2001.