HAL
open science

High-Accuracy Value-Function Approximation with Neural Networks Applied to the Acrobot

Rémi Coulom

To cite this version:

Rémi Coulom. High-Accuracy Value-Function Approximation with Neural Networks Applied to the Acrobot. 12th European Symposium on Artificial Neural Networks - ESANN'2004, Michel Verleysen, 2004, Bruges, Belgique, pp.7-12. inria-00107776

HAL Id: inria-00107776

https://inria.hal.science/inria-00107776v1

Submitted on 19 Oct 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

High-Accuracy Value-Function Approximation with Neural Networks Applied to the Acrobot

Rémi Coulom

CORTEX group, LORIA
Nancy, France

Abstract. Several reinforcement-learning techniques have already been applied to the Acrobot control problem, using linear function approximators to estimate the value function. In this paper, we present experimental results obtained by using a feedforward neural network instead. The learning algorithm used was model-based continuous TD(λ). It generated an efficient controller, producing a high-accuracy state-value function. A striking feature of this value function is a very sharp 4-dimensional ridge that is extremely hard to evaluate with linear parametric approximators. From a broader point of view, this experimental success demonstrates some of the qualities of feedforward neural networks in comparison with linear approximators in reinforcement learning.

1 Introduction

Reinforcement learning [9] is an adaptive technique to solve sequential decision problems, that is to say problems of selecting actions to control a process, in order to maximize some cumulative reward. This kind of technique has been applied successfully to domains such as game-playing and motor control.

Most of reinforcement-learning algorithms consist in evaluating a *value function* that estimates the outcome of acting from a particular state. When the system to be controlled can be in a very large number of states, this value function has to be estimated by a generalizing function approximator.

It is often advised that linear parametric function approximators are well adapted for this approximation task (linearity is meant as linearity of the output with respect to parameters), but non-linear function approximators such as feedforward neural networks can be used too. Linear approximators are preferred because of some theoretical convergence guarantees, and because of their ability to perform local generalization, which makes incremental learning more efficient.

In this paper, we demonstrate that feedforward networks can still perform better than linear approximators in terms of value-function accuracy on the acrobot swing-up task. The next section introduces notations and algorithms.

Section 3 presents experiment results. Section 4 discusses them and compares them to related work. The last section gives direction for further research.

2 Experimental Setting

2.1 General Formalism

A continuous reinforcement-learning problem is defined by:

- states $\vec{x} \in S \subset \mathbb{R}^p$,
- controls $\vec{u} \in U \subset \mathbb{R}^q$,
- system dynamics $f : S \times U \mapsto \mathbb{R}^p$, so that $\dot{\vec{x}} = f(\vec{x}, \vec{u})$,
- a reward function $r : S \times U \mapsto \mathbb{R}$,
- a shortness factor $s_\gamma \geq 0$ ($\gamma = e^{-s_\gamma \delta t}$).

A *strategy* or *policy* is a function $\pi : S \mapsto U$ that maps states to controls. Applying a policy from a starting state $\vec{x}_0$ at time t_0 produces a trajectory $\vec{x}(t)$ defined by the ordinary differential equation $\dot{\vec{x}} = f\big(\vec{x}, \pi(\vec{x})\big)$. The value function of π is defined by

$$V^\pi(\vec{x}_0) = \int_{t=t_0}^{\infty} e^{-s_\gamma (t-t_0)} r\Big(\vec{x}(t), \pi\big(\vec{x}(t)\big)\Big) dt \ . \tag{1}$$

The goal is to find a policy that maximizes the total amount of reward over time, whatever the starting state $\vec{x}_0$, that is to say π^* so that

$$\forall \vec{x}_0 \in S \quad V^{\pi^*}(\vec{x}_0) = \max_{\pi : S \mapsto U} V^\pi(\vec{x}_0) \ .$$

2.2 The Acrobot

The acrobot [7] is made of two articulated segments. One extremity (the "hands") is fixed, and the other (the "feet") is free. The control variable is a torque applied at the joint between the two segments. There are four state variables (θ_1, θ_2, $\dot{\theta}_1$, $\dot{\theta}_2$), which are the angles of the two segments with respect to the vertical axis, and their derivatives (θ_1 for "arms" and θ_2 for "legs"). The goal is to reach the vertical balance position. Reward is the height of the feet. Complete specifications are available in [2].

2.3 Learning Algorithm

In order to approximate the optimal value function V^* with a parametric function approximator $V_{\vec{w}}$, where $\vec{w}$ is the vector of weights, the continuous TD(λ) algorithm [3] consists in integrating an ordinary differential equation:

$$\begin{cases} \dot{\vec{w}} = \eta \mathcal{H} \vec{e} \ , \\ \dot{\vec{e}} = -(s_\gamma + s_\lambda)\vec{e} + \dfrac{\partial V_{\vec{w}}(\vec{x})}{\partial \vec{w}} \ , \\ \dot{\vec{x}} = f\big(\vec{x}, \pi(\vec{x})\big) \ , \end{cases} \tag{2}$$

with

$$\mathcal{H} = r\big(\vec{x}, \pi(\vec{x})\big) - s_\gamma V_{\vec{w}}(\vec{x}) + \frac{\partial V_{\vec{w}}}{\partial \vec{x}} \cdot f\big(\vec{x}, \pi(\vec{x})\big) \ .$$

$\mathcal{H}$ is the Hamiltonian and is a continuous equivalent of Bellman's residual. $\mathcal{H} > 0$ indicates a "good surprise" and causes and increase in the past values, whereas $\mathcal{H} < 0$ is a "bad surprise" and causes a decrease in the past values. The magnitude of this change is controlled by the learning rate η, and its time extent in the past is defined by the parameter s_λ. s_λ can be related to the traditional λ parameter in the discrete algorithm by $\lambda = e^{-s_\lambda \delta t}$. $\vec{e}$ is the vector of eligibility traces. Learning is decomposed into several episodes, each starting from a random initial state, thus insuring exploration of the whole state space. During these episodes, the policy π is chosen to be greedy with respect to the current value estimate $V_{\vec{w}}$. In this experiment, we used $\eta = 0.01$, $s_\lambda = 2$, trial length = 5s, Euler numerical integration with $\delta t = 0.02$s.

2.4 Function Approximators

The feedforward neural networks used in this experiments had fully-connected cascade architectures, with 30 neurons and 378 weights. Hidden neurons were sigmoids and the output was linear. Weights were initialized with a standard deviation equal to $1/\sqrt{m}$, where m is the number of connections feeding into the node [4]. Inputs were normalized and centered. For each angle θ, $\cos\theta$ and $\sin\theta$ were given as input to the network, to deal correctly with circular continuity. Learning rates were adapted as explained in [2], by simply dividing the learning rate of the output layer by the square root of the number of neurons.

3 Experiment Results

The acrobot was trained using the previous algorithm for 1 million trials. Progress was not monotonous: the TD(λ) algorithm has no convergence guarantees, and did indeed show some instabilities. Despite some short periods of performance decrease, the global trend was a steady performance improvement.

Figure 1 shows a trajectory of the acrobot after training, starting from the downward position. It managed to reach the vertical position at a very low velocity, but it could not keep its balance. Figure 2 shows a slice of the value function obtained.

4 Discussion and Related Work

Here is a summary of some of the most significant previous results obtained with reinforcement learning applied to the Acrobot:

- Sutton [8] managed to build an acrobot controller with the Q-learning algorithm using a tile-coding approximation of the value function. He used $u_{max} = 1$ Nm, and managed to learn to swing the endpoint of the

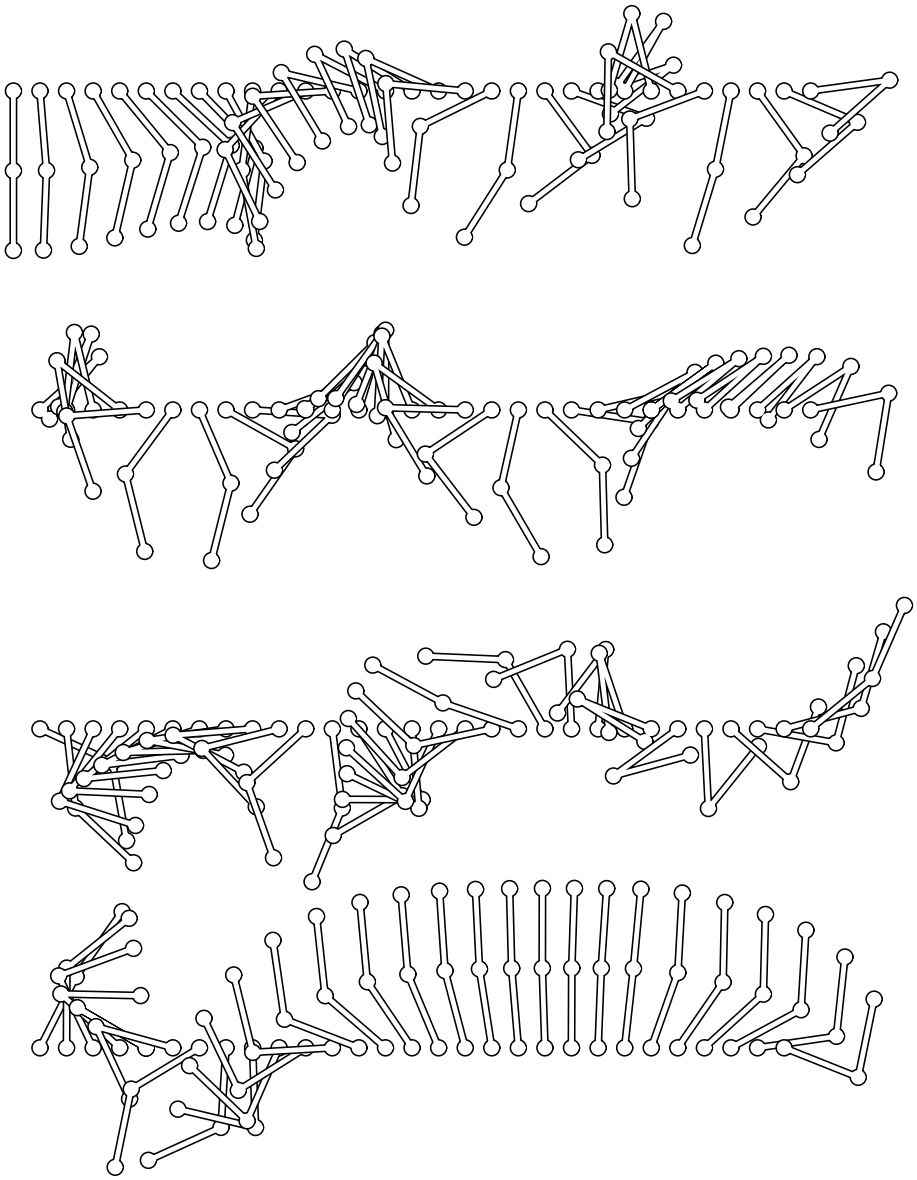

Figure 1: Acrobot trajectory. The time step of this animation is 0.1 seconds. The whole sequence is 12-second long.

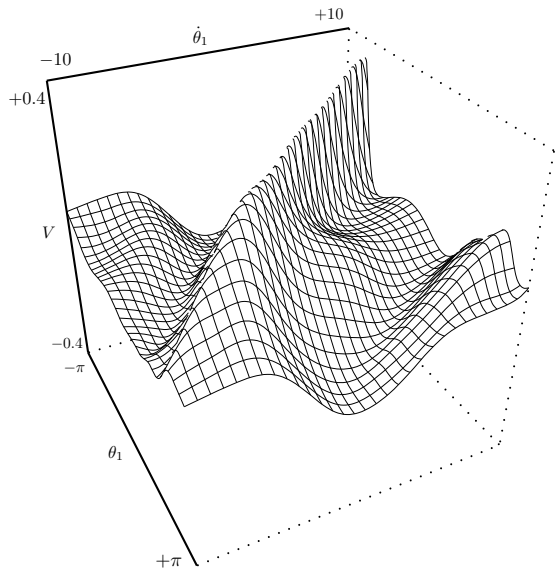

Figure 2: A slice of the Acrobot value function ($\theta_2 = 0$, $\dot{\theta}_2$=0)

acrobot above the bar by an amount equal to one of the links, which is a lot easier than reaching the vertical position.

- Munos [6] used an adaptive grid-based approximation trained by value iteration, and managed to teach the acrobot to reach the vertical position. He used $u_{max} = 2$ Nm (so did we), which makes the problem slightly easier. His goal was to reach the vertical position regardless of the velocity, so the acrobot could not keep its balance.
- Yoshimoto *et al.* [10] succeeded in balancing the acrobot with reinforcement learning. They used $u_{max} = 30$ Nm, which makes the problem much easier than with 1 or 2 Nm.
- Boone [1] obtained the best controllers, but the techniques he used are not really reinforcement learning (he did not build a value function) and are very specific of this kind of problem.

In comparison with previous work, experiments reported in this paper either solve a significantly more difficult problem (first three items), or use a more generic method (last item).

One particularly striking aspect of this result is the very sharp ridge of the value function plotted on Figure 2, approximated with a network of only 378 weights. Generic parametric linear function approximator would require a huge number of weights to get such a high resolution in a 4-dimensional state space.

The downside of this accuracy is the huge number of trials required. This make this method impractical for real-time learning on a real robot.

5 Conclusion and Perspectives

Experimental results demonstrated that a feedforward network can learn high-resolution features of a value function, at the expense of requiring a lot of trials. A challenge for the future would be to try to design algorithms that combine the data-efficiency of local linear approximators with the approximation ability of feedforward networks. A possibility might be to incorporate episodic memory into the reinforcement-learning process, to overcome the high interference [5].

References

[1] Gary Boone. Minimum-time control of the acrobot. In *1997 International Conference on Robotics and Automation*, pages 3281–3287, Albuquerque, NM, 1997.

[2] Rémi Coulom. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control*. PhD thesis, Institut National Polytechnique de Grenoble, 2002.

[3] Kenji Doya. Reinforcement learning in continuous time and space. *Neural Computation*, 12:243–269, 2000.

[4] Yann Le Cun, Leon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient BackProp. In Genevieve B. Orr and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*. Springer, 1998.

[5] James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995.

[6] Rémi Munos and Andrew Moore. Variable resolution discretization for high-accuracy solutions of optimal control problems. In *International Joint Conference on Artificial Intelligence*, 1999.

[7] Mark Spong. The swingup control problem for the acrobot. *IEEE Control Systems Magazine*, 15(1):49–55, February 1995.

[8] Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044. MIT Press, 1996.

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

[10] Junichiro Yoshimoto, Shin Ishii, and Masa-aki Sato. Application of reinforcement learning to balancing of acrobot. In *1999 IEEE International Conference on Systems, Man and Cybernetics*, volume V, pages 516–521, 1999.