Teaching Reinforcement Learning using a Physical Robot

Michel Tokic MICHEL.TOKIC@HS-WEINGARTEN.DE
Institute of Applied Research, University of Applied Sciences Ravensburg-Weingarten, Germany

Haitham Bou Ammar HAITHAM.BOUAMMAR71@GOOGLEMAIL.COM
Department of Knowledge Engineering, Maastricht University, The Netherlands

Abstract

This paper presents a little crawling robot as a didactic instrument for teaching reinforcement learning. The robot learns a forward-walking policy from scratch in less than 20 seconds of reinforced sensorimotor interactions. The state space consists of two discretized dimensions, where the behavior is visualizable and comprehensible. In laboratory tutorials, students conduct experiments with a Toolbox written in Java, enabling to vary significant parameters such as the exploration rate or the discounting factor. Our experience as well students feedback in using the crawler reflects: (1) a more interesting lecture, (2) an increase in students' motivation, and (3) an efficient and fast learning for students.

1. Introduction

Machine Learning is a fast growing field. Applications such as recommender and/or ranking systems are embedded in most search engines and social networking sites. Various other applications such as the *Google driverless car* and robotics, among others, are paving the way for new technologies and products in the near future.

Such applications raise a broad interest in learning about this new and exciting technology. For instance, over 100,000 students attending Andrew Ng's online lecture on Machine Learning clearly reflect this increasing interest. This interest elevation comes at the expense of providing the best tools to convey such a knowledge in an efficient and suitable manner. Machine learning is a massive field, which includes different areas of study. In an introductory course, typically few hours are assigned to reinforcement learning (RL), a technique for acquiring behavior based on reinforced sensorimotor interactions (Sutton and Barto, 1998). Aiming at conveying such a knowledge efficiently, we are looking for a vivid presentation of these various techniques. In particular we are looking for a simple to understand physical robot as a motivator for AI students (Kay, 2010). For starters and to better understand the delivered material, it is important to be able to visualize various aspects of the used algorithms, therefore, the learning problem should consist of a two dimensional state space. A robot fulfilling these requirements is the crawler (Kimura et al., 1997; Tokic et al., 2009) with its two degrees of freedom arm, shown in Figure 1. To couple the physical robot with a software kit, that allows the students to easily program a learning behavior, the open-source *Teaching-Box* has been introduced in former research (Ertel et al., 2009). This kit is implemented in Java and contains abstracted basic RL algorithms, allowing to design arbitrary experiments with this technology.

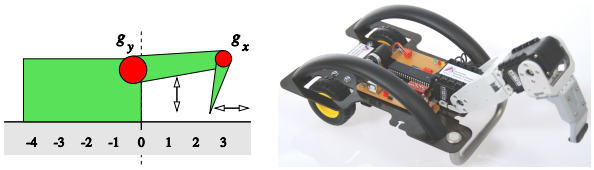
In this paper we reflect upon experience in using the crawler as an example of a physical demonstration in introductory AI courses (Ertel, 2011). It was clear that such a scheme has: (1) increased a students' interest and motivation to learn about reinforcement learning, and (2) made the lecture more exciting as there is a hands-on and easy to experiment-with example.

2. The Crawler and Teaching-Box

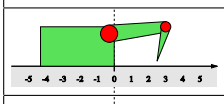
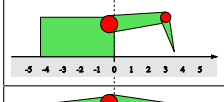
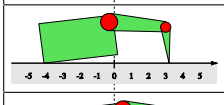
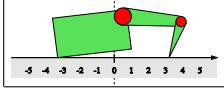
The physical robot, shown in Figure 1, is a two degrees of freedom arm. The state space is described using the discrete joint angles g_x and g_y shown in Figure 2. The goal of the robot is to move forward. Therefore, the reward is the speed of the robot, i.e. the distance the center of gravity of the robot had moved in a given time step. Consequently, a positive reward is attained if the robot moves forward and a

Appearing in *Proceedings of the Workshop on Teaching Machine Learning* at *29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

negative one vice versa. The robot is controlled by an ATmega32 micro-controller board mounted at its top as shown in Figure 1(b). The joints of the robot are driven by servos and the robot's movement–used to calculate the velocity–is measured using an optical encoder attached to one of the wheels. The board has outlets for the servos, a serial wireless transceiver, an outlet for the encoder, and a DIP switch to set various parameters.



(a) (b)

robot	time t	state g_y	state g_x	reward x	action a_t
	0	up	left	0	right
	1	up	right	0	down
	2	down	right	0	left
	3	down	left	1	up

(c)

Figure 1. (a) The robot model with its two joints and the (b) physical robot. Four steps of a simple forward-walking policy are shown in (c).

2.1. Reinforcement Learning

Reinforcement Learning (RL) is a technique for solving sequential decision making problems (Sutton and Barto, 1998). Typically, a learning problem is described as a Markov Decision Process (MDP) that basically consists of a state space, $\mathcal{S}$, an action space, $\mathcal{A}$, transition probabilities, $\mathcal{P}(s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ with $s \in \mathcal{S}$ and $a \in \mathcal{A}$, a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, and a discount factor $\gamma \in [0,1[$. The goal in RL is to learn a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$, with $\pi(a|s)$ defining the probability of choosing action a in state s. The goal of an RL agent is to learn π so as to maximize the total discounted reward, defined by an utility function, $V^\pi(s) = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right\}$.

Algorithm 1 ValueIteration On The Crawler

Initialize V arbitrarily, e.g., $V(s) = 0$, for all $s \in \mathcal{S}$
Initialize $\mathcal{R}^a_{ss'}$ arbitrarily, e.g., $r(s,a) = 0$, for all $r \in \mathcal{R}^a_{ss'}$
$state \leftarrow (g_x = 1, g_y = 1)$
loop
  $\xi \leftarrow \text{rand}(0..1)$
  if $\xi < \varepsilon$ **then**
    $a \leftarrow \text{rand}(\mathcal{A}(state))$
  else
    $a \leftarrow \text{argmax}_a \mathcal{R}^a_{ss'} + \gamma V(s')$
  end if
  $successorState \leftarrow \delta(state, a)$
  observe $r(state, a)$ and update $\mathcal{R}^a_{ss'}$
  for all $s \in \mathcal{S}$ **do**
    $V(s) \leftarrow \max_{a \in \mathcal{A}(s)} \Big[r(s,a) + \gamma V\big(\delta(s,a)\big) \Big]$
  end for
  $state \leftarrow successorState$
end loop

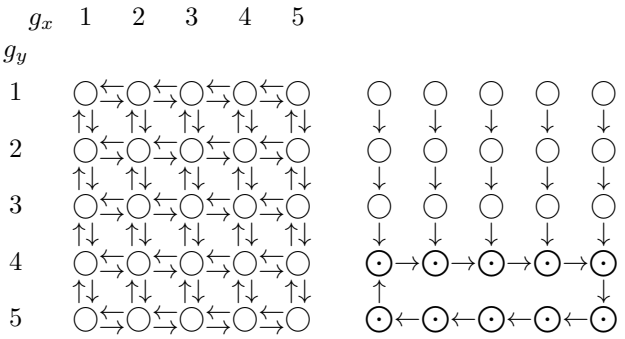


Figure 2. The cartesian 5×5 grid-world model (left) and a cyclic walking policy (right). States within the cycle are labeled as $\odot$.

2.2. Reinforcement Learning on the Crawler

There are different suggested algorithms for solving the crawler problem. In assignments and lab tutorials for the introduction to AI course, we use a simplified *Value Iteration* algorithm without using transition probabilities. This scheme is described in Algorithm 1, which starts by initializing all state values to arbitrary numbers such as zeros. Starting from the upper left corner, $state_{\text{init}} = \{g_x = 1, g_y = 1\}$, an action can be chosen according to an ε-greedy exploration policy. An action a performed in $state$ transitions the arm to a neighboring successor state, $state' \leftarrow \delta(state, a)$. The distance

the robot moved forward/backward is utilized as the reward signal r_{t+1}, which is measured as encoder tics after sending the action command to the servos. Subsequently the reward is memorized in the reward model $\mathcal{R}^a_{s,s'}$. Next, for all states the value iteration algorithm is performed using the so far sampled reward model $\mathcal{R}^a_{s,s'}$. Finally, the observed successor state, $state'$, is taken as the new actual *state* of the robot.

Students experimentation in this specific example is conducted by programming the previous algorithm within the Teaching-Box, explained in Section 2.3. The parameters γ and ϵ, as well as others, are allowed to vary so as to assess the behavior of the algorithm. This enables for a better understanding of RL and the overall problem.

2.3. Teaching-Box

In order to easily program RL algorithms for the crawler, we provide a simple Java software kit that allows: (1) test the efficacy of the algorithm in simulation, and (2) apply the algorithm on the real physical robot to see it running. The Teaching-Box (TB) is an easy to use Java RL library, which includes basic RL algorithms, e.g., Q-learning(λ) and SARSA(λ), as well function approximators such as CMACs or radial basis function networks. The TB is implemented in a way to easily allow students to directly vary parameters of the algorithms, generate plots, and other visualizations to better assess the quality of the learned behavior.

3. Class Room Experience

The crawler is used in AI classes for computer science master students in the second semester. For the laboratory tutorial lasting about 1.5 hours, basic knowledge of the Java programming language is required. At first, it is demonstrated how the crawler learns to walk forward in about 20 seconds. Further, the technicality and difficulty of the learning problem are explained as in Figure 1+2, by basically reflecting on the delayed feedback problem. This alone is able to raise interest for understanding the details of the problem. Then, the basic algorithms for solving MDPs are introduced, e.g., value iteration and Q-learning, which afterwards enables to start conducting experiments on the robot.

A typical task for a student is to observe and assess the robot's learning behavior on different surfaces, for instance by comparing the progress of the robot through examining the learned V and reward functions, or measuring the traveled distance in a fixed time interval. Another experiment is the variation of the algorithmic parameters such as the discount factor or the exploration rate. For instance, if the exploration rate is set to one, then the robot will perform 100% random actions. In contrast, if the exploration rate is constantly set to zero, the robot will be completely exploit the current V-values, which rarely leads to the optimal policy in practice, and to sub-optimal cycles instead, see Figure 3. This allows to better understand the *dilemma of exploration and exploitation* inherent to reinforcement learning. In other experiments, using for example the Q-learning algorithm, students observe a much slower learning behavior. This is because standard Q-learning updates only one Q-value after performing an action, but which might be improved by utilizing eligibility traces (Singh and Sutton, 1996) in simulation. However, on the hardware robot using the ATmega32 microcontroller, we only have 2kB of memory, for which reason we found the modified value iteration algorithm to perform best.

Due to the small state space and the idea of being able to visualize basically everything in the algorithm, students are usually highly motivated in starting to program the physical robot. Furthermore, a lot of interesting questions and discussions are initiated, for instance, extensions to continuous states and/or actions spaces are of major interest. Other questions on how to best choose the algorithmic parameters were also detailed and further analyzed. Starting from this simple example, some students even decided to further pursue bachelor and master theses in our groups.

4. Conclusions

In this paper we have presented an idea of using the simple crawling robot in teaching introductory machine learning courses. The robot is described by a 5×5 grid world, for which reason it is easy to understand and visualize. The hands-on physical experiment using the TB has raised the interest and motivation by students to better understand and analyze the subject, which allows to asses the behaviors of certain parameters involved in the algorithmic design process. Our experience with such type of learning is positive and we like to recommend using such a system at different universities all over the world. The software kit is open-source and available for anybody to use[1]. Demonstration videos and contact addresses for obtaining the robot are also available[2]. In the current SVN trunk, a beta release of a three dimensional visualization of the robot using PythonGL is also provided, which does not require to own the hardware robot. Besides this, the group of Jennifer S. Kay at

[1] http://sourceforge.net/projects/teachingbox
[2] http://www.tokic.com

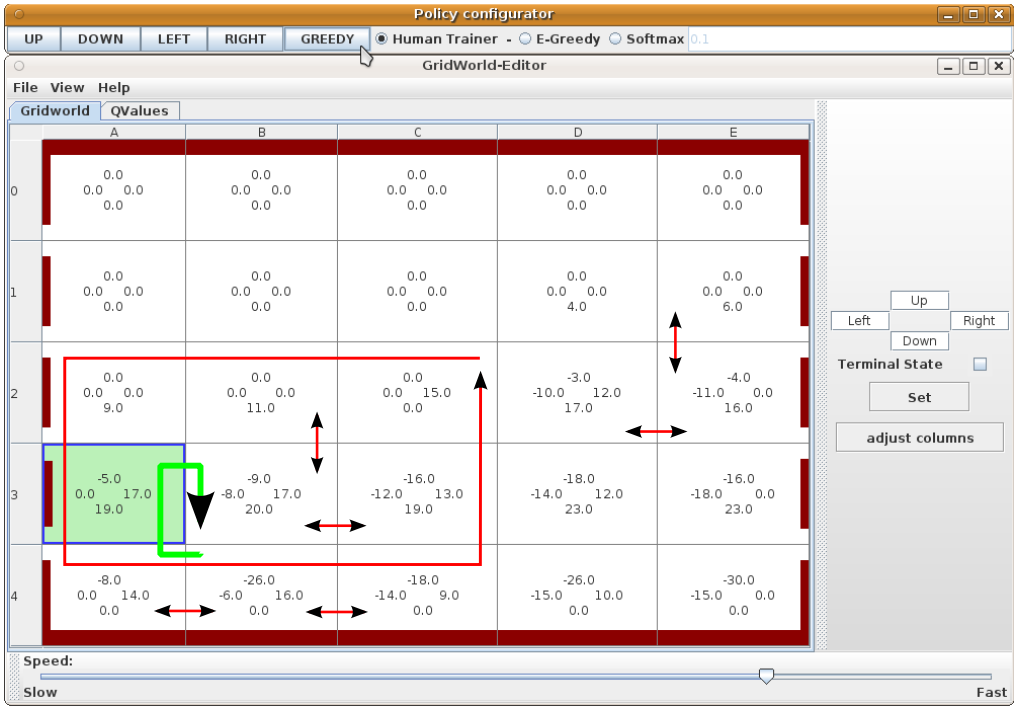


Figure 3. The grid-world editor of the Teaching-Box showing on top the ability to configure an action-selection policy. Numbers in cells indicate the reward $r(s, a)$ observed from the environment. The cycle A3→B3→B4→A4 indicates the optimal cycle having an average reward of $\bar{r} = \frac{17+20-6-8}{4} = 5.75$ per action. All other marked cycles are sub-optimal, i.e. having a lower average reward/action ratio compared to the optimal cycle. The cell with surrounded border (A3) indicates the current state of the robot.

Rowan University currently develops a version of the crawler utilizing LEGO Mindstorms (Montresor et al., 2011). Once finished, we plan to integrate it into the TB as well.

Acknowledgements

We like to thank Wolfgang Ertel and Jennifer S. Kay for sharing their experiences using the crawler.

References

W. Ertel. *Introduction to Artificial Intelligence*. Springer, 2011.

W. Ertel, M. Schneider, R. Cubek, and M. Tokic. The Teaching-Box: a universal robot learning framework. In *International Conference on Advanced Robotics ICAR'09*, pages 1–6, 2009.

J. S. Kay. Robots as recruitment tools in computer science: The new frontier or simply bait and switch? In *AAAI Spring Symposium: Educational Robotics and Beyond, Technical Report SS-10-03*. AAAI, 2010.

H. Kimura, K. Miyazaki, and S. Kobayashi. Reinforcement learning in POMDPs with function approximation. In *Proceedings of the 14th International Conference on Machine Learning (ICML'97)*, pages 152–160, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

S. Montresor, J. Kay, M. Tokic, and J. Summerton. Work in progress: Programming in a confined space - a case study in porting modern robot software to an antique platform. In *Proceedings of the 41st ASEE/IEEE Frontiers in Education Conference*, pages T3H–1–T3H–3, Rapid City, SD, USA, 2011. IEEE Press.

S. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158, 1996.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

M. Tokic, J. Fessler, and W. Ertel. The crawler, a class room demonstrator for reinforcement learning. In *Proceedings of the 22th International Florida Artificial Intelligence Research Society Conference*, pages 160–165, Menlo Park, USA, 2009. AAAI Press.