Learning of a ball-in-a-cup playing robot

Bojan Nemec, Matej Zorko, Leon Žlajpah
Robotics Laboratory, Jožef Stefan Institute
Jamova 39, 1001 Ljubljana, Slovenia
E-mail: bojan.nemec@ijs.si

***Abstract*—In the paper we evaluate two learning methods applied to the ball-in-a-cup game. The first approach is based on imitation learning. The captured trajectory was encoded with Dynamic motion primitives (DMP). The DMP approach allows simple adaptation of the demonstrated trajectory to the robot dynamics. In the second approach, we use reinforcement learning, which allows learning without any previous knowledge of the system or the environment. In contrast to the majority of the previous attempts, we used SASRA learning algorithm. Experimental results for both cases were performed on Mitsubishi PA10 robot arm.**

***Index Terms*—robot learning, trajectory generation, reinforcement learning.**

I. Introduction

A robot that learns to perform a task is one of the major challenges facing contemporary robotics. Robot learning by the demonstration relieves humans from tedious programming. Even more challenging are self-improving robots, that would allow operation in changeable or only partially known environments. Robot learning as a subpart of machine learning can be subdivided into two major categories - supervised learning and unsupervised learning.

Supervised learning is a machine learning technique for determining a model or a function from a set of training data. The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples. The resulting model can mimic the responses of a teacher who provides two sets of observations: inputs and the corresponding desired outputs. In contrary, in unsupervised learning an agent learns from interaction with its environment, rather than from a knowledgeable teacher that specifies the action the agent should take in any given state. Generally spoken, supervised methods assume the availability of some a priori knowledge, while unsupervised methods do not assume any a-priori knowledge. The computer has to learn from the data set by itself.

One of challenging tasks for the robot learning is ball-in-a-cup game, also known as Kendama, Balero or Bilboqet game. The aim of this game is to swing the ball hanging down on a string from a cup and catch it with the cup, as illustrated in Fig. 1. Ball-in-a-Cup has previously been studied by many researchers. Takenaka [1] used a three degree of freedom robot that is moving on a plane to perform Ball-in-a-cup. The robot was controlled by a trajectory tracking controller based on a mathematical model. The trajectory of the end effector was taken from human patterns, cameras were used to track the ball and the task was accomplished using the nonlinear compensations . Sakaguchi and Miyazaki [2], Sato et al. [3] took a two degree of freedom robot that receives feedback from force sensors, which corresponds to playing Ball-in-a-Cup with eyes closed. A model of the pendulum is given. The motion planning is divided in sub tasks. First, the robot performs some movements in order to estimate the parameters of the model. Finally, the trajectory is optimized based on this feedback. Miyamoto et al. [4] used a seven degree of freedom anthropomorphic arm. They record human motions which is used to train a neural network to reconstruct via-points. The imitation is designed to match the Cartesian coordinates exactly and the joints as closely as possible. The motion is performed open-loop and visual feedback is used to improve the trajectory. Shone et al. [5] construct a simple one degree of freedom robot particularly for Ball-in-a-Cup. The trajectory is found using a gradient based path planner in simulation. Ball-in-a-Cup is modeled as pendulum switching to a free point mass. The found trajectory is executed on the real system, which only has the motor encoders as feedback. Fujii et al. [6] modeled Kendama using the approximation of string as line segments, Arisumi et al. [7] use a casting manipulator to throw the ball on the spike. A new approach based on Natural Actor Critic (NAC) was proposed also by Kobler and Peters [8], [9]. They used Barret WAM arm and a vision system for this task. Based on previously demonstrated trajectory the system was able to learn appropriate trajectory in approximately 75 rollouts.

In the paper, we apply two basic types of robot learning to the ball-in-a-cup game. In the first approach we use imitation learning as a representative of the supervised learning category. In the second approach we use reinforcement learning as a representative of the unsupervised learning category. In this later approach we applied SARSA learning algorithm combined with the eligibility traces.

II. Learning ball-in-a-cup by demonstration

Unlike in for an example tennis swing, where a human just needs to learn a goal function for the one moment the racket hits the ball, in Ball-in-a-Cup we need a complete dynamical system as cup and ball constantly interact. A straightforward approach is to mimicking the human motion by recording a sequence of movements and reproduce the same motion by the robot. Due to different dynamics capabilities of the robot and humans this approach fails as a rule. The

Fig. 1. Ball-in-cup

trajectory to be executed by the robot has to be appropriately modified. A central issue in trajectory generation is the choice of the representation (or encoding) of the trajectory. The simplest representations is storing trajectories in large time-indexed vectors. Alternatively, more compact representations can be constructed by using interpolation algorithms such as spline-fitting and other parameterized representation. Other approaches use neural networks for function-approximators. Among the must suitable for use in robotics are function aproximators using dynamical systems. Here, we will use Dynamic Motion Primitives (DMP), proposed by Ijspeert [10].

A. Encoding trajectories with dynamic motion primitives

In the DMP formulation, motion in each task coordinate is represented as a damped mass-spring system perturbed by an external force. Such a system can be modeled with a set of differential equations [11]

$$\begin{aligned} \dot{v} &= \frac{1}{\tau}\left(K(g-y) - Dv + f(x)\right), \qquad (1) \\ \dot{y} &= \frac{v}{\tau} \end{aligned}$$

where v and y are the velocity and position of the system, g is the goal position of the trajectory, x is the phase variable, which defines the time evolution of the trajectory, τ is the temporal scaling factor, K is the spring constant, and D is the damping. The phase variable x is defined by

$$\dot{x} = -\frac{\alpha x}{\tau}. \qquad (2)$$

For trajectory generation it is necessary that the dynamic system is critically damped and thus reaches the goal position without overshoots. A suitable choice is $D = 2\sqrt{K}$, where K is chosen to meet the desired velocity response of the system. Function $f(x)$ is a nonlinear function which is used to adapt the response of the dynamic system to an arbitrary complex movement. A suitable choice for $f(x)$ was proposed by Ijspeert et al. [10] in the form of a linear combination of M radial basis functions

$$f(x) = \frac{\sum_{j=1}^{M} w_j \psi_j(x)}{\sum_{j=1}^{M} \psi_j(x)} x, \qquad (3)$$

where ψ_j are Gaussian functions defined as

$$\psi_j(x) = \exp(-\frac{1}{2\sigma_j^2}(x - c_j)^2).$$

Parameters c_j and σ_j define the center and the width of the j-th basis function, while w_j are the adjustable weights used to obtain the desired shape of the trajectory.

B. Motion Acquisition

The trajectory represented by a second order dynamic system is parameterized with the initial velocity and position, the final goal position, and a set of weights w_j associated with radial basis functions. In this section we present the procedure for the calculation of weights w_j. We assume that we obtain trajectory data points $q_d(t_i), \dot{q}_d(t_i), \ddot{q}_d(t_i),\ t_i \in [0, T]$ from human demonstration. Next, we define function f^* as

$$f^*(t) = \tau^2 \ddot{y} + \tau D \dot{y} + K(y - g), \qquad (4)$$

where $y = y(t)$. Our task is to find a set of weights w_j that minimize the quadratic cost function

$$J = \sum_{i=0}^{N} (f^*(t_i) - f(x(t_i)))^2. \qquad (5)$$

We use global regression methods to find the optimal weights w_j. It results in the following linear system of equations

$$\mathbf{A}\mathbf{w} = \mathbf{f}^*, \qquad (6)$$

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_M \end{bmatrix}, \quad \mathbf{f}^* = \begin{bmatrix} f^*(t_0) \\ \vdots \\ f^*(t_N) \end{bmatrix},$$

$$\mathbf{A} = \begin{bmatrix} \frac{\psi_1(x(t_0))x(t_0)}{\sum_{j=1}^{M}\psi_j(x(t_0))} & \cdots & \frac{\psi_M(x(t_0))x(t_0)}{\sum_{j=1}^{M}\psi_j(x(t_0))} \\ \vdots & \vdots & \vdots \\ \frac{\psi_1(x(t_N))x(t_N)}{\sum_{j=1}^{M}\psi_j(x(t_N))} & \cdots & \frac{\psi_M(x(t_N))x(t_N)}{\sum_{j=1}^{M}\psi_j(x(t_N))} \end{bmatrix}.$$

In order to decrease the computational burden, it is favorable to compute a solution to (6) recursively by incrementally updating the following quantities

$$\mathbf{P}_i = \mathbf{P}_{i-1} - \frac{\mathbf{P}_{i-1}\mathbf{a}_i\mathbf{a}_i^T\mathbf{P}_{i-1}}{1 + \mathbf{a}_i^T\mathbf{P}_{i-1}\mathbf{a}_i}, \qquad (7)$$

$$\mathbf{w}_i = \mathbf{w}_{i-1} + (f^*(t_i) - \mathbf{a}_i^T\mathbf{w}_{i-1})\mathbf{P}_i\mathbf{a}_i, \qquad (8)$$

where $\mathbf{a}_i$ is the M dimensional column vector associated with the corresponding row of the matrix $\mathbf{A}$ and the optimal weights are $\mathbf{w} = \mathbf{w}_N$.

The benefit of using DMP-s in our case is the simplicity in the way how the learned trajectories can be adjusted when executed by the robot. In our case it turned out that the swing up procedure can be tuned with the single parameters τ, which is the temporal scaling factor. Similarly, catching position was encoded with a single parameter g, which determines the goal position. Due to the dynamic properties of DMP-s we can change parameters even during the trajectory execution.

C. Predicting the catching pose

In order to catch the ball, we have to predict the ball trajectory. Ball position is captured by a stereo vision system. The free flying ball trajectory can be described with a set of equations, $x(t) = x_0 + v_{x0}t$, $y(t) = y_0 + v_{y0}t + g\frac{t^2}{2}$ and $z(t) = z_0 + v_{z0}t$ with six unknown parameters $x_0, y_0, z_0, v_{x0}, v_{y0}$ and v_{z0}. x_0, y_0, z_0 and v_{x0}, v_{y0}, v_{z0} are initial position and initial velocities of the free flying object, respectively, and g denotes gravitation acceleration here (see Fig 2 for the coordinate system notation). Rearranging them into the form $x(t) = [1 \quad t][x_0 \quad v_{x0}]^T$, $y(t) + g\frac{t^2}{2} = [1 \quad t][y_0 \quad v_{y0}]^T$ and $z(t) = [1 \quad t][z_0 \quad v_{z0}]^T$, we can use similar set of equations as for DMP weights estimation (Eq. 6) and predict the ball coordinates $x(t)$, $y(t)$ and $z(t)$ recursively. Clearly, we have an infinite number of choices where to catch the ball. In our experiment we choose the catching position that requires minimal accelerations to move from the current robot position to the goal position.

III. Reinforcement Learning of Ball-in-a-Cup

In this section we attempt to learn swing-up motion for the ball-in-a-cup without any previous knowledge of the system or the environment. We decided to evaluate SARSA-learning algorithm for this problem [12]. All previous studies used the function approximation reinforcement learning algorithms [13], [9]. A common believe is that the SARSA-learning is less appropriate for solving such a problem because it is hard to assure the Markov decision propriety [12]. The modified algorithm for the SARSA learning with eligibility traces [12] is described by the following Eq.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]e(s)$$

$$e_t(s) = \left\{ \begin{array}{cc} \gamma\lambda e_{t-1}(s) & if s \neq s_t \\ \gamma\lambda e_{t-1}(s) + 1 & if s = s_t \end{array} \right\}, \qquad (9)$$

where Q is the is the action-value function and s_t and a_t are the state and action of the agent at the time t, respectively. α and γ denote the learning rate and discount rate; these two parameters determine the learning rate and the convergence, respectively. r_t denotes reward associated with the agent state, obtained in the current time t. The second part of the Eq. 9 describes the exponential decay of the eligibility traces. Each time the agent is in a state s_t, the trace for that state is reset to 1. Eligibility traces are a bridge between the Monte-Carlo methods and the temporal-difference method. In our case they help the agent to remember faster good actions. One of the key properties of the learning system is how we assign states s and actions a. Best results were obtained by selecting states composed of the cup position x, cup velocity $\dot{x}$, angle between the cup and the ball φ and angular velocity $\dot{\varphi}$, as show in the Fig 2. For SARSA learning, states had to be discretized. Cup position was described with two values, cup velocity with three values, ball angle with 18 values spanning over the entire circle and the ball angular velocity with 3 values, forming all together a set of 324 values. Action value was chosen to be cup acceleration $\ddot{y}$ described with 5 values spanning from maximal negative to maximal positive acceleration. Commonly, ϵ-greedy method is used for learning. In our case we obtained much faster learning with a random set of harmonic trajectories, used as input trajectory in the entire rollout. Rollouts with the learning trajectories were randomly selected, similar as in the ϵ-greedy method. The learning goal was to swing the ball to the desired angle $\varphi(t_0)$ with the desired angular velocity $\dot{\varphi}(t_0)$, which results in zero radial acceleration at $t = t_0$. The behavior of the learning system is determined mostly by the reward assigned to current state. The final state was rewarded with a positive reward. Also, during the swing up, each state was rewarded depending on the swing angle. One of the major problem of swing-up learning is the limited displacement of the cup in x direction due to the limited workspace of the robot. To prevent the robot from reaching its limits, excessive displacement x was penalized by a negative reward.

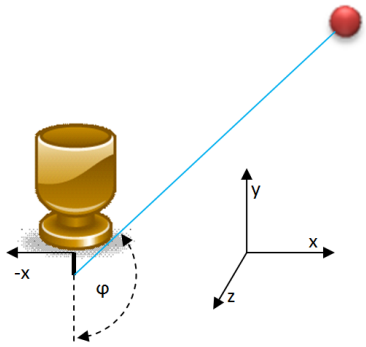

Fig. 2. Angles notation for ball-in-cup

IV. Experimental Results

Experimental evaluation of a robot playing ball-in-a-cup was implemented using Mitsubishi Pa10 robot with 7 D.O.F. The ball trajectory was tracked with a stereo vision system at 120 Hz [14]. Human motion trajectory of the game was captured using the Smart motion capture system at 100 Hz. During the learning, only the swing up procedure was of our interest. Catching was performed based on prediction of the ball free-flight trajectory, as described in subsection II-C. The captured trajectory was encoded with DMP using 30 kernel functions ($M = 30$). As expected, the robot failed to swing up the ball to the desired height using the captured trajectory from the human demonstrator. It turned out that that the swing up procedure can be tuned with the single parameters τ, which determines the execution time of the trajectory. After the ball has reached the desired height, new goal g for the DMP was calculated using the procedure described in subsection II-A. The ball trajectory and the robot trajectory are displayed in Fig. 3, respectively. Figure 4 displays a robot pose during the ball-in-a-cup playing.

In the next experiment we evaluated reinforcement learning for the swing-up procedure of the same task. The learning was

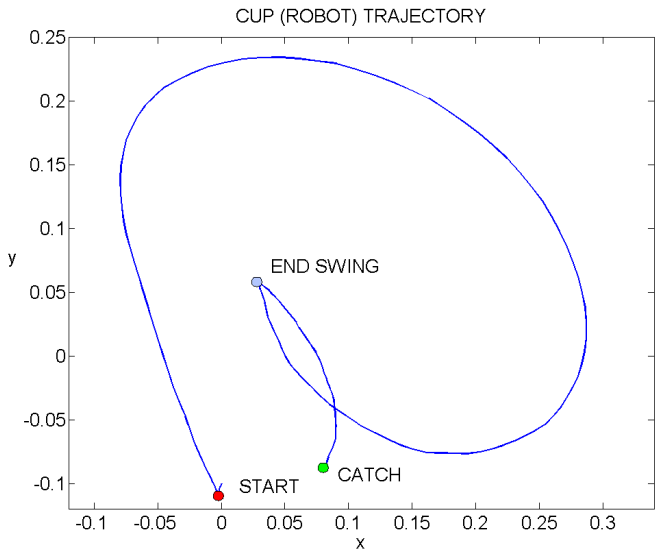

Fig. 3. Cup trajectory during the ball-in-a-cup game

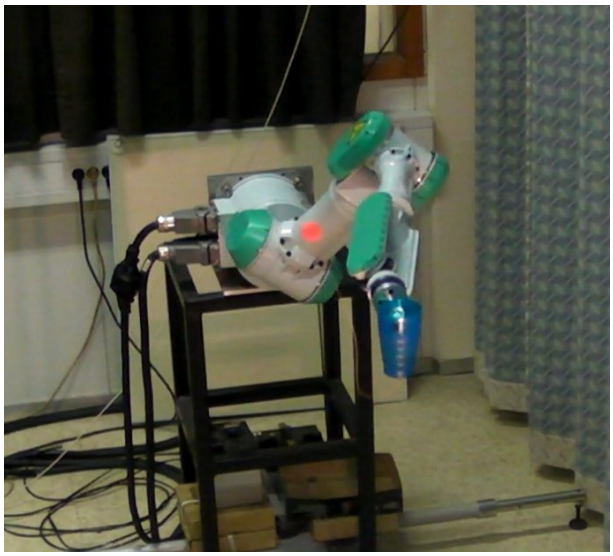

Fig. 4. Robot during the ball-in-a-cup game

performed without any initial knowledge of the system. Initial learning was performed on the simulation system. Using the algorithm described in the section III, 220 to 300 rollouts were required to learn the swing-up trajectory. Figure 5 shows one example of obtained rewards during the learning. As we can see from the Fig. 5, learning ends after approx. 260 rollouts, the robot repeats the learned trajectory and gets constant reward. Figures 6 and 7 show ball trajectory, learned robot trajectory and a simulated robot pose after the successful learning, respectively. Learning was then continued on the real robot, using the final value of the action value matrix Q, learned in the simulation, as an initial action value matrix in the real robot. The real robot was able to adjust the swing-up trajectory in 40 to 90 rollouts. Note that the robot does not necessarily learns equal swing-up trajectory if we repeat leaning procedure again, because and infinite number of trajectories can swing-up the ball to the desired height. Note also that the learned trajectory is completely different from the human demonstrated trajectory (Fig 3). Human demonstrator used only one swing to swing-up the ball and the swing-up trajectory is in the $x - y$ plane. In contrast to that, the robot was instructed to learn the swing up trajectory by moving x coordinate only. Due to the restricted accelerations, the learned trajectory required two swings in order to swing-up the ball to the desired height.

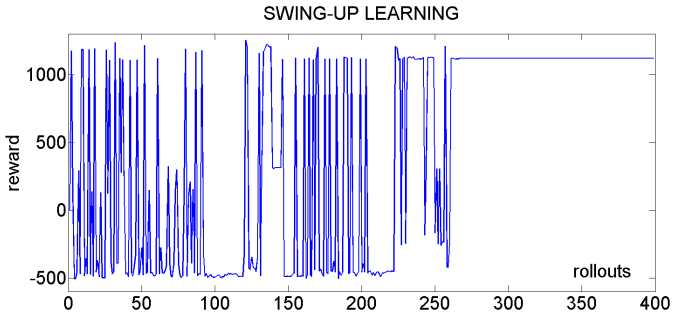

Fig. 5. Reward during swing-up learning

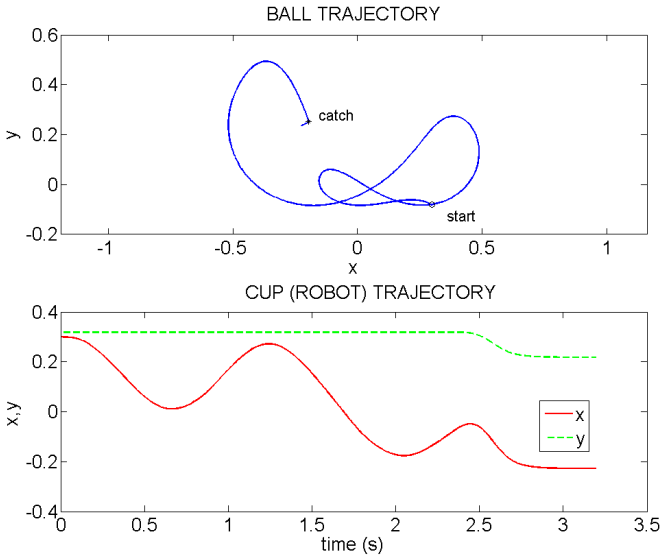

Fig. 6. Learned ball and swing-up trajectory

V. Conclusion

In the paper, we evaluate two learning methods for playing ball-in-a-cup game with a robot. First method is a representative of the supervised learning methods. We use the dynamic motion primitives to encode the human demonstrated trajectory and adapt it to the robot dynamic. The second approach uses reinforcement learning without any previous knowledge of the system. In the paper it is shown, that the SARSA learning is appropriate also for the trajectory learning, providing an appropriate selection of the system states. Both

Fig. 7. Simulated robot during the ball-in-a-cup game

approaches were experimentally evaluated using the Mitsubishi Pa10 robot.

REFERENCES

[1] K. Takenaka. *Dynamical control of manipulator with vision : cup and ball game demonstrated by robot.* Transactions of the Japan Society of Mechanical Engineers. C, 50(458):2046-2053, 1984.

[2] T. Sakaguchi and F. Miyazaki. *Dynamic manipulation of ball-in-cup game.* In Proceedings of the International Conference on Robotics and Automation (ICRA), :2941-2948, 1994.

[3] S. Sato, T. Sakaguchi, Y. Masutani, and F. Miyazaki. *Mastering of a task with interaction between a robot and its environment : kendama task.* Transactions of the Japan Society of Mechanical Engineers. C, 59(558):487-493, 1993.

[4] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato. *A kendama learning robot based on bi-directional theory.* Neural Networks, 9(8):1281-1302, 1996.

[5] T. Shone, G. Krudysz, and K. Brown. *Dynamic manipulation of kendama.* Technical report, Rensselaer Polytechnic Institute, 2000.

[6] T. Fujii, T. Yasuda, S. Yokoi, and J.-i. Toriwaki. *A virtual pendulum manipulation system on a graphic workstation.* In Proceedings of the 2nd IEEE International Workshop on Robot and Human Communication (RO-MAN), 1993.

[7] H. Arisumi, K. Yokoi, and K. Komoriya. *Kendama game by casting manipulator.* In Proceedings of the IEEE/RSJ 2005 International Conference on Intelligent Robots and Systems (IROS), 2005.

[8] J. Kobler. *Reinforcement Learning for Motor Primitives.* Diplomarbeit, Universitt Stuttgart, 2008,

[9] J. Kobler, J. Peters. *Policy Search for Motor Primitives in Robotics.* in Neural Information Processing Systems (NIPS), 2008

[10] A. J. Ijspeert, J. Nakanishi, and S. Schaal, *Movement imitation with nonlinear dynamical systems in humanoid robots.* in Proc. IEEE Int. Conf. Robotics and Automation, Washington, DC, :1398–1403, 2002.

[11] S. Schaal, P. Mohajerian, and A. Ijspeert. *Dynamics systems vs. optimal control – a unifying view.* Progress in Brain Research, 165(6):425–445, 2007.

[12] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press, 1998.

[13] K. Doya. *Reinforcement Learning In Continuous Time and Space.* Neural Computation, 12(1):219-245, 2000.

[14] A. Ude, T. Shibata, C.G Atkeson. *Real-time visual system for interaction with a humanoid robot.* Robot. auton. syst. , 37(1):115-125, 2001.