Autonomous Reinforcement Learning with Experience Replay for Humanoid Gait Optimization

Paweł Wawrzyński

Institute of Control and Computation Engineering, Nowowiejska 15/19, 00-665 Warsaw, Poland

Abstract

This paper demonstrates application of Reinforcement Learning to optimization of control of a complex system in realistic setting that requires efficiency and autonomy of the learning algorithm. Namely, Actor-Critic with experience replay (which addresses efficiency), and the Fixed Point method for step-size estimation (which addresses autonomy) is applied here to approximately optimize humanoid robot gait. With complex dynamics and tens of continuous state and action variables, humanoid gait optimization represents a challenge for analytical synthesis of control. The presented algorithm learns a nimble gait within 80 minutes of training.

Keywords: reinforcement learning, autonomous learning, learning in robots.

1. Introduction

In contemporary economic and technical reality control systems work as programmed by human designers. Their expertise is expensive and only suffices to solve problems of limited complexity. The field of Reinforcement Learning (RL) was founded to create an alternative: a set of techniques for control systems to learn, mainly by trial-and-error, proper behaviour instead of being programmed. Such techniques are potentially attractive both because they could limit the cost of control system synthesis but also solve practical control problems intractable for human expert analysis.

Earlier works [1, 2] present a combination of Actor-Critic reinforcement learning with experience replay and step-size estimation by means of fixed-point algorithm. This approach is applied here to optimize humanoid robot gait. The present paper touches upon the following issues.

- Scalability and applicability to continuous domains both assured by Actor-Critic scheme [3, 4, 5, 6, 7].
- **Efficiency** which results from experience replay [8, 1, 9] i.e., storing data on agent-environment interaction in a database, calling them in a process simultaneous to the interaction, and using them for Actor and Critic updates as if the events described by the data has just happened.
- **Autonomy.** As many other RL schemes, Actor-Critic with experience replay is based on incremental adjustments of policy parameters with the use of improvement direction estimates. It is therefore a stochastic approximation procedure [10] and works only when it is provided with lengths of improvement steps i.e., it requires step-sizes. Proper values of step-sizes generally depend on the problem and the stage of the process. Their estimation on-the-fly has found to be a difficult problem and its general solution has not been found over several decades of research. Among many approaches to step-size estimation [11, 12, 13, 14, 15], the one applied here is the fixed-point algorithm [16] since it does not need any problem-dependent parameters and thus provides real autonomy.
- **Bipedal gait synthesis.** Control of bipedal walking is usually synthesised with analytic methods like Zero Moment Point [17], or Central Pattern Generators [18]. Reinforcement learning has also been tried for that purpose [19, 20, 21], but only to solve a certain subproblem in policy optimization.

In this paper reinforcement learning is applied to optimize control of bipedal walking in realistic scenario in which:

- The system to be controlled is difficult to model (and simulate), and is not exercised at all.
- However, there exists a certain common-sense based controller that makes the system work, albeit very clumsily.
- The role of reinforcement learning is to approximately optimize the controller, in a relatively short training, autonomously enough not to require tuning of parameters by the experimenter.

This paper is organized as follows. In Sec. 2 the problem of humanoid gait optimization is described. Another section put this problem in the frame of reinforcement learning. Sec. 4 overviews the particular RL algorithm applied to the problem. Sec. 5 presents application of the algorithm to the problem and the last section concludes.

### 2. Problem formulation

The problem at hand is to make a given humanoid robot walk, or possibly run, as fast as possible. Fig. 1 presents the robot. It is built of the following parts:

1. A body of Bioloid\(^1\). It is composed of 18 servomotors: 6 in each leg and 3 in each arm. It is 35 cm tall and weights 2 kg.
2. A backpack containing a small yet fully operational PC with Linux on board as well as an inertial sensor (ADIS 16365, an accelerometer and gyroscope in a single chip\(^2\)).
3. Feet with 4 touch sensors each.

The control goal is to make the robot walk straight as fast as possible under the following circumstances.

1. The terrain is even.
2. Information on the robot state only comes from servomotors’ encoders (positions of the servos), the inertial sensor (acceleration and angular velocity), and the touch sensors.
3. There exists a certain initial reactive policy that controls walking of the robot. This policy alone makes the robot follow a certain handcrafted cyclic trajectory.

The robot control policy is a reactive one. Every 33 ms the state of the robot is transformed into target positions of its servos. These positions are the sum of two components: the one resulting from the initial policy and the one resulting from one that is a subject of on-line optimization. The latter initially produces zeros and therefore at the very beginning the robot is only controlled by the predefined policy.

The predefined policy is based on a cyclic trajectory in the configuration space of the robot. Current state of the robot is projected onto this trajectory. The point that is on the trajectory 33 ms after the projection is determined and it becomes a vector of target positions of the robot servomotors. The role of the learning policy is to incrementally modify those positions.

---

\(^1\)Bioloids are serially manufactured by Robotis: [www.robotis.com](http://www.robotis.com).

\(^2\)ADIS 16365 is manufactured by Analog Devices: [www.analog.com](http://www.analog.com).
Fig. 1. The robot.
3. Reinforcement Learning

Reinforcement learning offers solutions to learning control problem under the Markov Decision Process (MDP) framework [22]. The setup concerns an agent that observes the state of its environment, $s_t$, in discrete time, $t = 1, 2, 3, \ldots$, performs an action, $a_t$, which moves the environment to the next state, $s_{t+1}$, and gives the agent a reward, $r_t \in \mathbb{R}$. The environment is in general stochastic which means that the consecutive state, $s_{t+1}$, is a result of sampling from the transition distribution conditioned on the preceding state, $s_t$, and the action, $a_t$, i.e.,

$$s_{t+1} \sim P_s(\cdot | s_t, a_t).$$

The reward may depend deterministically on the current action and the next state, $r_t = r(a_t, s_{t+1})$. A particular MDP is a tuple $\langle S, A, P_s, r \rangle$ where $S$ and $A$ are the state and action spaces, respectively; $\{P_s(\cdot | s, a) : s \in S, a \in A\}$ is a set of state transition distributions; and $r$ is the reward function. The transition distributions, $P_s$, and the reward function, $r$, are initially unknown to the agent. The goal of learning is to determine a stochastic control policy that assigns actions to states such that in each state the agent may expect the highest rewards in the future.

In this paper the typical statement of reinforcement learning goal is considered. Namely, the actions are selected randomly by the policy defined as a probability distribution parameterized by the state and the policy vector, $\theta \in \mathbb{R}^n$, which can be represented as

$$a \sim \pi(\cdot ; s, \theta).$$

(1)

The distribution, $\pi$, with a fixed parameter, $\theta$, defined a policy $\pi_\theta$.

The objective of learning is to set the policy vector such that the value-function,

$$V^{\pi_\theta}(s) = E \left( \sum_{t \geq 0} \gamma^t r_{t+1} | s_t = s, \text{ policy } = \pi_\theta \right)$$

(2)

is maximized for all states, $s$. The parameter $\gamma \in [0, 1)$ is the discount factor that defines the weight of distant rewards in relation to those obtained sooner.

Requirements

In this paper, we understand the agent as the part of the robot controller, whose operation is to be optimized in real time. We seek a learning algorithm with the following characteristics.

- It operates in continuous and multidimensional $S$ and $A$ spaces. Their discretization is not possible.
- It is autonomous i.e., does not depend on manually tuned parameters.
- It is fast due to efficient exploitation of data. The control process is slow in relation to computational power, and there are sufficient resources to store the agent experience and intensive computation thereof.

4. Autonomous Actor-Critic with experience replay

This section overviews the algorithm presented in [2] and its usage for stochastic control policy optimization. The algorithm operates in the following framework.

- **Control process.** It works in discrete time $t = 1, 2, \ldots$. At each instant, an action, $a_t$, is selected by the stochastic control policy (1), and applied to the process. The value $\pi_t = \pi(a_t; s_t, \theta)$, the resulting reward, $r_t$, and the following state, $s_{t+1}$, are registered, and the quintet

$$\langle s_t, a_t, \pi_t, r_t, s_{t+1} \rangle$$

(3)

is put in a database.

- **Optimization process.** It is based on the actor-critic framework, with Actor being the stochastic control policy and Critic that is a neural approximator of the value function with weights vector, $\nu$. The process updates the policy parameter, $\theta$, and Critic parameter, $\nu$, on the basis of the data collected in the database. The policy parameter is uploaded to the control process.
A single update performed by the optimization process is based on the following points:

- A sequence of consecutive quintets (3), starting from \( i \)-th, is selected randomly from the database.
- An vector, \( \hat{\phi}_i \), is computed to estimate the direction in which the policy parameter has to be incremented for the policy to produce better actions from the state \( s_i \). The policy update takes the form
  \[
  \theta \leftarrow \theta + \beta^\theta \hat{\phi}_i,
  \]
  where \( \beta^\theta \) is the policy vector step-size.
- An vector, \( \hat{\psi}_i \), is computed that estimates the direction in which Critic parameter has to incremented for the critic to play its role better in state \( s_i \). This vector update takes the form
  \[
  \upsilon \leftarrow \upsilon + \beta^\upsilon \hat{\psi}_i,
  \]
  where \( \beta^\upsilon \) is Critic’s step-size.

Performing incremental updates on the basis of improvement direction estimators locates this approach in the class of stochastic approximation procedures [10]. A procedure of that type is only autonomous if it is defined along with a way to compute its step-sizes; otherwise they may be defined experimentally but this contradicts the autonomy. The method of step-size estimation applied here is based on the fixed-point algorithm [16]. The general idea in this approach is as follows. The process is divided into parts in which the step-size remains constant. In each part, the improvement directions are computed both for the moving parameter and for the fixed one, that has began the current part. At the end of the part the step-size is updated on the basis of discrepancy between the sums of improvement estimates computed both ways: if the discrepancy is too small, it means that the step-size has been too small and is being increased. Conversely, if the discrepancy is too large, it means that the step-size has been too large as well, and it is being decreased.

5. Experimental results

The training of the robot is divided into episodes lasting for about 15 sec. in which the robot is able to travel about 1 meter. It is given rewards that includes the following components: (i) large negative penalty if the robot...
has just fallen, (ii) a moderate penalty for turning, (iii) a reward for difference of speed of forward movement of the elevated leg and the supporting one. The speed of the whole robot forward movement is not rewarded directly. Its estimation is possible but it would require the use of the kinematic model of the robot contrary to the assumption that the controlled system is not modelled thus remaining a black box.

The learning algorithm has the form applied in [2] for Half-Cheetah. Both Actor and Critic are here based on two layered feedforward neural networks with 200 neurons in hidden layers.

Figure 2 presents the results of training in the form of learning curve: average rewards vs. trial no. The robot learns quite soon to move faster than initially and the rewards increase from the very beginning. Simple increase of the speed of movement results in falling down, for which the robot is severely penalised. This is manifested by downward picks in the learning curve, especially in the first 50 episodes. Having gathered this traumatic experience, the robot learns to move forward fast while keeping balance. Therefore after 50 first episodes the frequency of falling down decreases and the average rewards increase.

Initially the robot walked with the speed of 5.1 cm/s. The initial policy was the fastest possible in the sense that increasing velocity on the same path would only result in instability and falling down of the robot. After 300 episodes of training the speed reached 11.1 cm/s. During the experiment the robot walked for about 80 minutes. However, the experiment lasted more, about two hours, because of the necessity to cool the servos which were getting overheated. The experiment was repeated several times with very similar results.

6. Conclusions

In this paper, an actor-critic reinforcement learning algorithm with experience replay and autonomously estimated step-sizes was applied to humanoid robot gait optimization. It was shown that the algorithm obtained over twofold increase in speed within 80 minutes of training without deterioration of stability of movement.

The scenario of training was corresponding to one in which reinforcement learning is meant to work: (i) the control problem with multidimensional state and action spaces, and complex dynamics requires much time and resources when solved analytically, (ii) the learning algorithm is efficient enough to provide good control in reasonable time before the controlled system gets destroyed by inappropriate actions of the learning controller, (iii) the learning process does not require repetitions because it is autonomous: it only depends on parameters that can be assigned by an experimenter basing on his experience (like the discount factor) or parameters that are estimated on the fly (the step-sizes).

References