Reinforcement Learning with Particle Swarm Optimization Policy (PSO-P) in Continuous State and Action Spaces

Daniel Hein, Technische, Universität München, Munich, Germany
Alexander Hentschel, Siemens AG, Munich, Germany
Thomas A. Runkler, Siemens AG, Munich, Germany
Steffen Udluft, Siemens AG, Munich, Germany

ABSTRACT

This article introduces a model-based reinforcement learning (RL) approach for continuous state and action spaces. While most RL methods try to find closed-form policies, the approach taken here employs numerical on-line optimization of control action sequences. First, a general method for reformulating RL problems as optimization tasks is provided. Subsequently, Particle Swarm Optimization (PSO) is applied to search for optimal solutions. This Particle Swarm Optimization Policy (PSO-P) is effective for high dimensional state spaces and does not require a priori assumptions about adequate policy representations. Furthermore, by translating RL problems into optimization tasks, the rich collection of real-world inspired RL benchmarks is made available for benchmarking numerical optimization techniques. The effectiveness of PSO-P is demonstrated on the two standard benchmarks: mountain car and cart pole.

KEYWORDS

Benchmark, Cart Pole, Continuous Action Space, Continuous State Space, High-dimensional, Model-based, Mountain Car, Particle Swarm Optimization, Reinforcement Learning

INTRODUCTION

Reinforcement learning (RL) is an area of machine learning inspired by biological learning. Formally, a software agent interacts with a system in discrete time steps. At each time step, the agent observes the system’s state \( s \) and applies an action \( a \). Depending on \( s \) and \( a \), the system transitions into a new state and the agent receives a real-valued reward \( r \in \mathbb{R} \). The agent’s goal is to maximize its expected cumulative reward, called return \( R \). The solution to an RL problem is a policy, i.e. a map that generates an action for any given state.

This article focuses on the most general RL setting with continuous state and action spaces. In this domain, the policy performance often strongly depends on the algorithms for policy generation and the chosen policy representation (Sutton & Barto, 1998). In the authors’ experience, tuning the policy-learning process is generally challenging for industrial RL problems. Specifically, it is hard to assess whether a trained policy has unsatisfactory performance due to inadequate training data, unsuitable policy representation, or an unfitting training algorithm. Determining the best problem-specific RL approach often requires time-intensive trials with different policy configurations and
training algorithms. In contrast, it is often significantly easier to train a well-performing system model from observational data, compared to directly learning a policy and assessing its performance.

To bypass the challenges of learning a closed-form RL policy, the authors adapted an approach from model-predictive control (Rawlings & Mayne, 2009; Camacho & Alba, 2007), which employs only a system model. The general idea behind model-predictive control is deceptively simple: given a reliable system model, one can predict the future evolution of the system and determine a control strategy that results in the desired system behavior. However, complex industry systems and plants commonly exhibit nonlinear system dynamics (Schaefer, Schneegass, Sterzing, & Udluft, 2007; Piche, et al., 2000). In such cases, closed-form solutions to the optimal control problem often do not exist or are computationally intractable to find (Findeisen & Allgoewer, 2002; Magni & Scattolini, 2004). Therefore, model-predictive control tasks for nonlinear systems are typically solved by numerical on-line optimization of sequences of control actions (Gruene & Pannek, 2011). Unfortunately, the resulting optimization problems are generally non-convex (Johansen, 2011) and no universal method for tackling nonlinear model-predictive control tasks has been found (Findeisen, Allgoewer, & Biegler, 2007; Rawlings, Tutorial overview of model predictive control, 2000). Moreover, one might argue based on theoretical considerations that such a universal optimization algorithm does not exist (Wolpert & Macready, 1997).

The main purpose of the present contribution is to provide a heuristic for solving RL problems which employs numerical on-line optimization of control action sequences. As an initial step, a neural system model is trained from observational data with standard methods. However, the presented method also works with any other model type, e.g., Gaussian process or physical models. The resulting problem of finding optimal control action sequences based on model predictions is solved with Particle Swarm Optimization (PSO), because PSO is an established algorithm for non-convex optimization. Specifically, the presented heuristic iterates over the following steps. (1) PSO is employed to search for an action sequence that maximizes the expected return when applied to the current system state by simulating its effects using the system model. (2) The first action of the sequence with the highest expected return is applied to the real-world system. (3) The system transitions to the subsequent state and the optimization process is repeated based on the new state (go to step 1).

As this approach can generate control actions for any system state, it formally constitutes an RL policy. This Particle Swarm Optimization Policy (PSO-P) deviates fundamentally from common RL approaches. Most methods for solving RL problems try to learn a closed-form policy (Sutton & Barto, 1998). The most significant advantages of PSO-P are the following. (1) Closed-form policy learners generally select a policy from a user-parameterized (potentially infinite) set of candidate policies. For example, when learning an RL policy based on tile coding (Sutton, 1996), the user must specify partitions of the state space. The partition’s characteristics directly influence how well the resulting policy can discriminate the effect of different actions. For complex RL problems, policy performances usually vary drastically depending on the chosen partitions. In contrast, PSO-P does not require a priori assumptions about problem-specific policy representations because it directly optimizes action sequences. (2) Closed-form RL policies operate on the state space and are generally affected by the curse of dimensionality (Bellmann, 1962). Simply put, the number of data points required for a representative coverage of the state space grows exponentially with the state space’s dimensionality. Common RL methods, such as tile coding, quickly become computationally intractable with increasing dimensionality. Moreover, for industrial RL problems it is often very expensive to obtain adequate training data prohibiting data-intensive RL methods. In comparison, PSO-P is not affected by the state space dimensionality because it operates in the space of action sequences.

From a strictly mathematical standpoint, PSO-P follows a known strategy from nonlinear model-predictive control: employ on-line numerical optimization to search for the best action sequences. While model-predictive control and RL target almost the same class of control-optimization problems with different methods, the mathematical formalisms in both communities are drastically different. Particularly, the authors find that the presented approach is rarely considered in the RL community.
Short-Term Generation Scheduling Solved with a Particle Swarm Optimizer
[www.igi-global.com/chapter/short-term-generation-scheduling-solved/72833?camid=4v1a](www.igi-global.com/chapter/short-term-generation-scheduling-solved/72833?camid=4v1a)

Transmission Expansion Planning by using DC and AC Models and Particle Swarm Optimization
[www.igi-global.com/chapter/transmission-expansion-planning-using-models/72832?camid=4v1a](www.igi-global.com/chapter/transmission-expansion-planning-using-models/72832?camid=4v1a)