Reinforcement Learning in Robotics: A Survey

...

Composition by Jens Kober, J. Andrew Bagnell, and Jan Peters
Presented by Dahlman, Jensen, and Dahlman

Overview

Adapting theoretical reinforcement learning techniques to robotics

Challenges of reinforcement learning in robotics

Increasing tractability of reinforcement learning

Examples of reinforcement learning in robotics

Interplay between MAS and robotics

- Behavior generation through RL in robots

Reinforcement Learning + Robotics

Reinforcement Learning

Provides feedback in terms of a scalar objective function

- Measures and evaluates a step's performance

Contains a state, action, and policy function

Goal is to determine (near) optimal policy, ensuring maximum total reward

- Maximum average reward in a continuous sense

RL vs Other Learning Types

Versus supervised learning

- No explicit instructions given to an agent
- Doesn’t have guaranteed rewards upon completion

Versus imitation learning

- No “expert” agent or protocol to follow example

RL builds upon complexity

- Still depends on a semi-defined system of states, actions
- Makes no assumption on completeness of system or accuracy of knowledge
- Useful in incomplete, continuous, intractable situations

What is Reinforcement Learning (RL)?

Details of RL

Maximization of reward over a period

Policy π to optimize reward gained for state-action pair

Action = π(state)

Action = P(action | state)

Rewards

R(s)

R(s, a)

R(s, a, s`)

Focused on a specific time H, or collectively discounted by some gamma factor

Further Contrast

Exploration

- Understand the system's available states, actions, and potential optimality
- Exploration-Exploitation Tradeoff

Dimensionality

- Realistic problem spaces are generally too large - curse of dimensionality
- Must use Lagrangian Multipliers
  - Value function V*
    - The optimal policy
  - State-Action function Q*

Three RL Methods

Dynamic Modeling

Require explicit data about transition to model reward - simplistic

Monte Carlo - Rollout

Frequency of transitions and rewards create approximation of V* - no model needed

Temporal-Based

Update V* value at each time step, based on current and future expected reward (Q-Learning)

RL Tradeoffs

Dynamic Modeling

Simple & cheap

Naive models, significant time to optimality

Monte Carlo - Rollout

Focus on optimal policy, rather than V* = faster optimality

Brittle and computationally hard

Expensive, difficult to implement

Temporal-Based

Cheaper and easier than MC-R

Challenges in Robot Reinforcement Learning

Curse of Dimensionality

Curse of Real-World Samples

Curse of Under-Modeling and Model Uncertainty

Curse of Goal Specification

Curse of Dimensionality

“As the number of dimensions grows, exponentially more data and computation are needed to cover the complete state-action space.”

Curse of Real-World Samples

- Robots require expensive hardware, will experience wear and tear and need maintenance
  - If you want to use reinforcement learning, you have to safely explore to reduce need for repairs
  - Safe exploration is often neglected by learning community
- Properties of the robot can change due to real-world conditions
  - Learning process may actually never converge
- Difficult to pinpoint external factors
  - Comparing learning algorithms is difficult because external factors aren’t clear
- Noise and inability to measure all states

The Real-World Causes A Plethora of Problems

- Since real-world samples are expensive in time, labor, and finances, it is often considered more important in robotic reinforcement learning to limit real-world interaction time
  - Not memory consumption or computational complexity, which is opposite to typical robotic development
- The real-world is continuous, digital computers are not
  - Time-discretization can cause distortion and other undesirable effects
- Robots can’t take action instantaneously
  - Slight delay between a measurement and the action results in observations based on the action being several time stamps later

Curse of Under-Modeling and Model Uncertainty

Idea: Offset the cost of actually interacting with the real-world by using a simulation

“As small model errors due to this under-modeling accumulate, the simulated robot can quickly diverge from the real-world system.”

Curse of Goal Specification

In traditional reinforcement, desired behavior is specified by a reward function and the goal is to maximize the long term reward

Monitoring the variance in the reward signal to improve leads to **reward shaping**

Help guide the system to the desired behavior with intermediate rewards and penalties

"Reward shaping gives the system a notion of closeness to the desired behavior instead of relying on a reward that only encodes success or failure."

Inverse optimal control is an alternative to manual reward function development

Instead of trying to build a function to get a desired effect, it is easier to demonstrate the appropriate behavior to the robot

Tractability Through Representation, Prior Knowledge, and Models

How to Fail at Learning in Robotics

“Naive application of reinforcement learning techniques in robotics is likely to be doomed to failure.”

How to Win at Learning in Robotics

“The remarkable successes… have been achieved by leveraging a few key principles – effective representations, approximate models, and prior knowledge of information.”

Tractability Through Representation

Making reinforcement learning methods tractable is tightly coupled to the underlying framework, including smart state-action discretization, value-function approximation, and pre-structured policies

Smart State-Action Discretization

Hand Crafted Discretization

For low dimensional tasks, you can generate the discretizations by splitting up the dimensions

Challenge: Find the right number of regions to split the dimension into

Learned from Data

Build the discretizations while learning

Meta-Actions

Perform more intelligent actions that are really just a sequence of movements that complete a simple task

Relational Representations

Value-Function Approximation

Physics-inspired Features

If hand-crafted features are known, value function approximation can be done using a linear combination of features

Neural Networks

Various different network approaches have been applied to robot rein learning,

Neighbors

Generalize from neighboring cells instead of looking at each local instance and applying learning to global network

Local Models

Pre-structured Policies

- Via Points & Splines
  - Open-loop policies (only dependent on time)
- Linear Models
- Motor Primitives
  - Combine linear models for dynamics with movement parametrizations
- Neural Networks
- Controllers
- Non-parametric

Tractability Through Prior Knowledge

Using prior knowledge can significantly reduce the exploration space and therefore speeds up the learning process. Ways to incorporate prior knowledge include initial policies, demonstrations, and models or constraints on the policy.

Prior Knowledge Through Demonstration

Entities learn through demonstrations from an expert

Common technique: apprenticeship learning

- Learning from a teacher and by practice

Key benefits of learning through demonstration:

- Provides supervised training of what actions to perform in certain states
- Removes need for global exploration; robot now knows which states are important

Common ways to acquire knowledge

- Demonstrations by a teacher

Prior Knowledge Through Task Structuring

Hierarchical Reinforcement Learning

- Decompose task into hierarchical basic components
- Doesn’t assume that all levels are fixed but learned all simultaneously

Progressive Learning

- Decompose task into sequence of increasingly difficult tasks
- Inspired by how biological systems work

Tractability Through Models

Using model-based learning, as opposed to model-free methods discussed earlier, can also make RL more tractable. Model-based RL involves learning forward models and mental rehearsal.

Issues in Mental Rehearsal

Simulation bias

- Policy that works well in simulation, but poorly in the real-world scenario
- Similar to over-fitting in supervised learning

Randomness of the real world

- Can explicitly model the uncertainty that naturally exists with each state and action

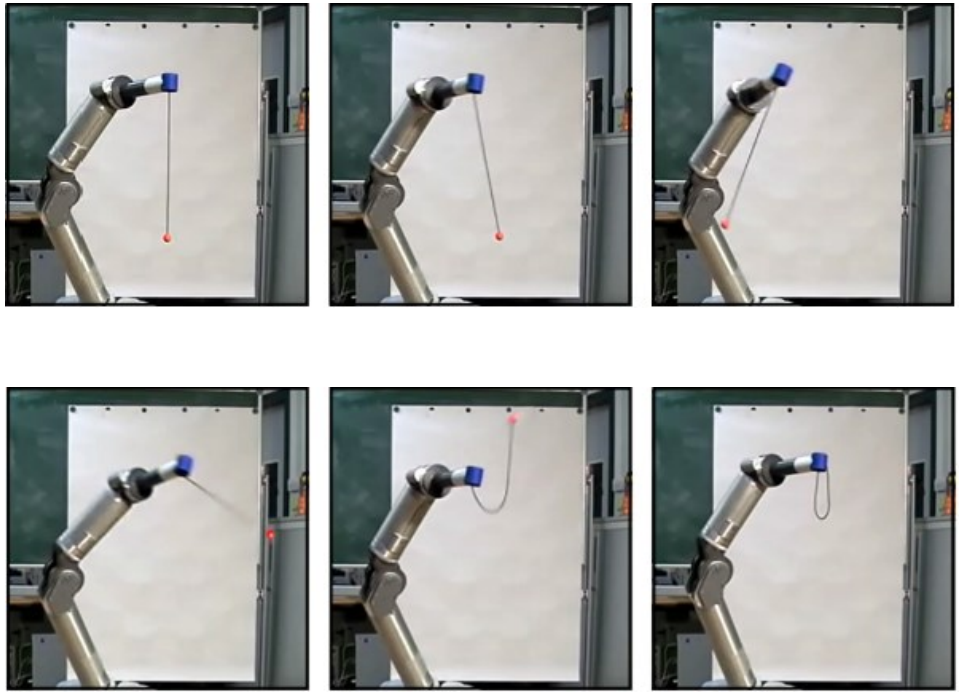
Optimization when sampling from a simulator

- Learning forward models are often trained by sampling random numbers
- Must find a way to negate the detriments of large vs small sample size

Successful Learning Approaches with Forward Models

- Iterative learning control
  - Create initial approximate model
  - Evaluate model
  - Update to reflect new knowledge, and loop back to first step
- Locally Linear Quadratic Regulators
  - Directly compute optimal control policies
- Value Function Methods with Learned Models
  - Pretend simulated model is generated from the real system

Ball-in-a-cup Case Study



System Description

States:

Robot joint angles

Robot joint velocities

Actions

Joint space accelerations

20 state and 7 action dimensions

Reward function

Binary

Reward if ball lands in cup, no reward otherwise

Notion of closeness

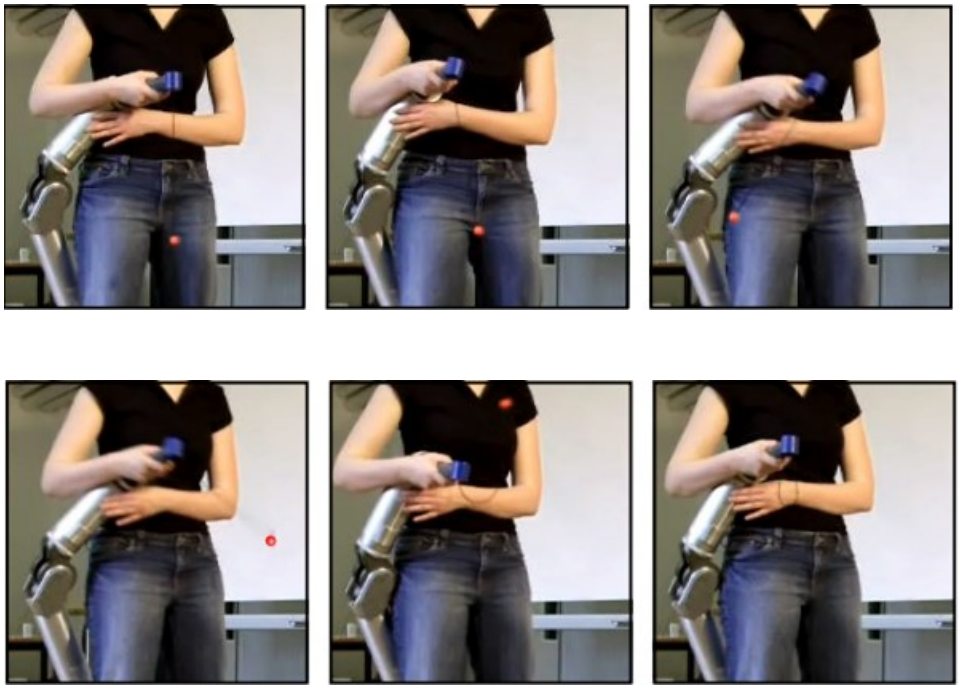
Minimal distance between ball and cup

Tweaked so that ball must pass above rim of cup

Policy Representation

“The resulting movement can start from arbitrary positions and velocities and go to arbitrary final positions while maintaining the overall shape of the trajectory.”

Teacher Demonstration



Policy Search

- Task is episodic
  - Optimization of the initial teacher demonstration is good enough to lead to a solution
  - Continuous states
  - Continuous actions
  - Many dimensions

Simulation

- Pendulum that switches to ballistic point mass
- Simulation policies didn’t quite work for real world
  - Usually missed by several centimeters
- Saved significant time

Results with Real Robot

Used algorithm derived from simulation, tuned the algorithm based on the results of each episode in the real environment

Ball first lands in cup: 42-45 episodes

Regular success: 70-80 episodes

Converged to maximum: after 100 episodes

Tested with two different movement strategies

- Swinging ball
- Pulling ball straight up

Discussion

Open questions, challenges, and lessons for robotic reinforcement learning

Open Questions

RL hasn’t been as researched as other methods, e.g. supervised learning

Requires a lot of research to choose a RL method

All methods require hand-tuning

Can we automate the choice of RL elements?

- How do we choose representations?

Develop more robust algorithms to limit tuning?

- How would we generate reward functions?
- How much prior knowledge is useful?

Challenges

Future research needed to solve problems such as:

- Reuse learned information
  - Could transfer learned skills to other tasks
  - Could share learned skill with other robots
- Performing large scale experiments
  - Hardware is fragile and expensive
  - Many differences between hardware, standardization is in progress

Lessons from Robotic Experiences for RL

Robotic RL takes different approaches from RL theory

- Robotic problems are multi-dimensional, require continuous actions and constant adaptation

Almost always prior knowledge about system

- Constructing even very crude models yields better results

Difference in reward systems in traditional RL vs robotic RL

- Classical RL: discrete rewards
- Robotic RL: reward-shaping with physically motivated rewards, continuous values, and multiple objectives

Questions?

Kober, J; Bagnell, D.; Peters, J. (2013). Reinforcement Learning in Robotics: A Survey, International Journal of Robotics Research, 32(11), pp. 1236-1272.