Reinforcement Learning in Continuous Environments

64.425 Integrated Seminar: Intelligent Robotics

Oke Martensen

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

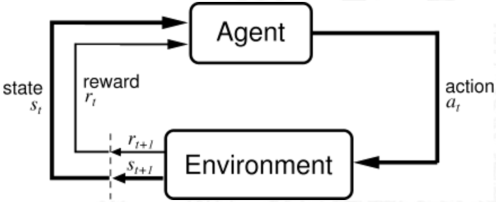
30. November 2015

Outline

Classical Reinforcement Learning

Agent := algorithm that learns to interact with the environment.
Environment := the world (including actor)

Goal:
optimize agent's
behaviour wrt.
a reward signal.



Sutton and Barto (1998)

Problem as Markov Decision Process (MDP): *(S, A, R, T)*

The General Procedure

Policy $\pi :=$ action selection strategy

- exploration and exploitation trade-off
- e.g. ϵ-greedy, soft-max, ...

Different ways to model the environment:

- value functions $V(s)$, $Q(s, a)$: cumulative discounted reward expected after reaching state s (and after performing action a)

Standard Algorithms

Sutton and Barto (1998)

Temporal-difference (TD) learning

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$$

Numerous algorithms are based on TD learning:

- SARSA
- Q-Learning
- actor-critic methods (details on next slide)

Actor-Critic Models

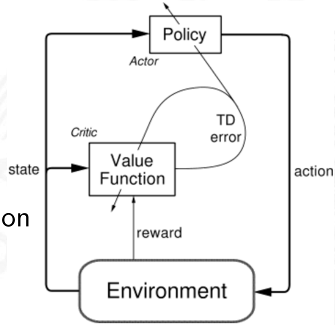
A TD method with separate memory structure to explicitly represent the policy independent of the value function.

Actor: policy structure

Critic: estimated value function

The critic's output, TD error, drives all the learning.

- computationally cheap action selection
- biologically more plausible



Sutton and Barto (1998)

Why is RL so Cool?

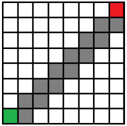
- it's how humans do
- sophisticated, hard-to-engineer behaviour
- can cope with uncertain, noisy, non-observable stuff
- no need for labels
- online learning

"The relationship between [robotics and reinforcement learning] has sufficient promise to be likened to that between physics and mathematics" Kober and Peters (2012)

The Continuity Problem

So far: discrete action and state spaces.
Problem: world ain't discrete.

Example: moving on a grid world



Continuous state spaces have already been investigated a lot.

Continuous action spaces, however, remain a problem.

Tackling the Continuity Problem

1. Discretize spaces, then use regular RL methods
   - e.g. tile coding: group space into binary features receptive fields
   - But: How fine-grained? Where to put focus? Bad generalization ..
2. Use parameter vector $\vec{\theta}_t$ of a function approximator for updates
   - often neural networks are used and the weights as parameters

CACLA — Continuous Actor Critic Learning Automaton

Van Hasselt and Wiering (2007)

- learns undiscretized continuous actions in continuous states
- model-free
- computes updates and actions very fast
- easy to implement (cf. pseudocode next slide)

CACLA Algorithm

Algorithm Cacla

Given γ, an initial state distribution I and an MDP to act on.
Initialize $\vec{\theta}$, $\vec{\psi}$, $s \sim I$.
repeat
  Choose $a \sim \pi(s, \vec{\psi})$
  Perform a, observe r and s'
  $\delta = r + \gamma V(s') - V(s)$
  $\vec{\theta}^T = \vec{\theta}^T + \beta \delta \nabla_\theta V(s)$
  if $\delta > 0$ **then**
    $\vec{\psi}^T = \vec{\psi}^T + \alpha(a - Ac(s, \vec{\psi}))\nabla_\psi Ac(s, \vec{\psi})$
  end if
  if s' is terminal **then**
    $s \sim I$
  else
    $s = s'$
  end if
until end

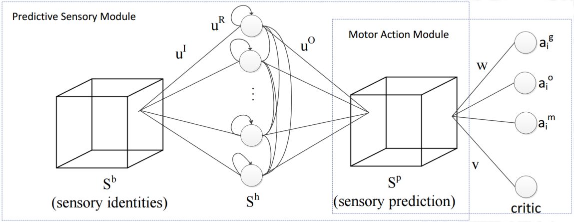
$\vec{\theta}$: parameter vector
$\vec{\psi}$: feature vector

Van Hasselt (2011)

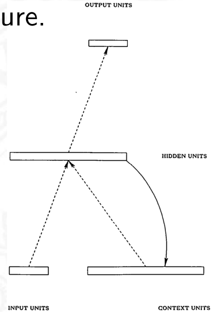
A bio-inspired model of predictive sensorimotor integration
Zhong et al. (2012)

Latencies in sensory processing make it hard to do real time robotics; noisy, inaccurate readings may cause failure.

1. Elman network for sensory prediction/filtering
2. CACLA for continuous action generation



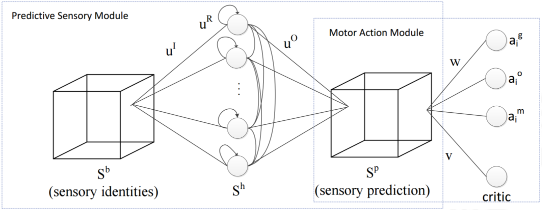
Zhong et al. (2012)



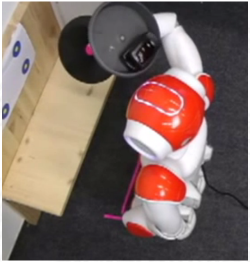
Elman (1990)

Robot Docking & Grasping Behaviour

Zhong et al. (2012)



Zhong et al. (2012)



https://www.youtube.com/watch?v=vF7u18h5IoY

- more natural and smooth behaviour
- flexible wrt. changes in the action space

Conclusion

Challenges:

- problems with high-dimensional/continuous states and actions
- only partially observable, noisy environment
- uncertainty (e.g. *Which state am I actually in?*)
- hardware/physical system:
  - tedious, time-intensive, costly data generation
  - reproducibility

Solution approaches:

- partially observable Markov decision processes (POMDPs)
- use of filters: raw observations + uncertainty in estimates

Thanks for your attention!

Questions?

References

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Kober, J. and Peters, J. (2012). Reinforcement Learning in Robotics: A Survey. In Wiering, M. and van Otterlo, M., editors, *Reinforcement Learning*, volume 12, pages 579–610. Springer Berlin Heidelberg, Berlin, Heidelberg.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Van Hasselt, H. and Wiering, M. (2007). Reinforcement learning in continuous action spaces. In *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on*, pages 272–279. IEEE.

Van Hasselt, H. P. (2011). *Insights in reinforcement learning*. Hado Van Hasselt.

Zhong, J., Weber, C., and Wermter, S. (2012). A predictive network architecture for a robust and smooth robot docking behavior. *Paladyn, Journal of Behavioral Robotics*, 3(4):172–180.