Reinforcement Learning in Continuous State and Action Spaces

Lucian Bușoniu
Advisers: Robert Babuška, Bart De Schutter

Delft University of Technology
Center for Systems and Control
Project: Interactive Collaborative Information Systems

13 January 2009
Motivation and focus

Learning can find solutions that:
- are hard to determine a priori
- improve over time

Reinforcement learning:
- uses reward signal as performance feedback
- can work without prior knowledge

Exact RL solutions only in discrete cases:
- this thesis: continuous spaces
- using approximate solutions
Motivation and focus

Learning can find solutions that:
- are hard to determine a priori
- improve over time

Reinforcement learning:
- uses reward signal as performance feedback
- can work without prior knowledge

Exact RL solutions only in discrete cases:
- this thesis: continuous spaces
- using approximate solutions
Motivation and focus

**Learning** can find solutions that:
- are hard to determine a priori
- improve over time

**Reinforcement learning:**
- uses reward signal as performance feedback
- can work without prior knowledge

**Exact RL solutions** only in discrete cases:
- this thesis: continuous spaces
- using approximate solutions
Outline

1. Reinforcement learning
2. Challenge & contribution
3. Examples
4. Conclusions
Optimal control problem

Example: robot should move to goal in shortest time
Elements of RL

- Robot in given state (position $X$, $Y$)
- Robot takes action (e.g., move forward) and reaches new state
- Receives reward = quality of state transition
Elements of RL

- Robot in given state (position $X$, $Y$)
- Robot takes action (e.g., move forward) and reaches new state
- Receives reward = quality of state transition
Elements of RL

- Robot in given state (position $X$, $Y$)
- Robot takes action (e.g., move forward) and reaches new state
- Receives reward = quality of state transition
Elements of RL

- Robot in given state (position $X, Y$)
- Robot takes action (e.g., move forward) and reaches new state
- Receives reward = quality of state transition
Policy

- **Control policy**: what action to take in every state

  \[ u = h(x) \]

- E.g., in state \([9, 1]\), move **forward**
  in state \([2, 10]\), move **right**
Policy

- **Control policy**: what action to take in every state
  \[ u = h(x) \]

  - E.g., in state \([9, 1]\), move **forward**
  - in state \([2, 10]\), move **right**
Performance criterion

- **Reward** = one-step performance
- **Return** = long-term performance, along trajectory

\[ \begin{align*}
R(x, u) &= r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \ldots \\
0 < \gamma < 1 & \text{ discount factor}
\end{align*} \]
**Performance criterion**

- **Reward** = one-step performance
- **Return** = long-term performance, along trajectory

\[
R(x, u) = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \ldots
\]

- $0 < \gamma < 1$ discount factor
Goal: obtain maximal return:
\[ R^*(x, u) = r_1 + \text{discounted rewards} \]
along best trajectory starting in \( x_1 \)

Optimal policy \( h^* \) can be computed from \( R^* \)
Optimality

- Goal: obtain maximal return:
  \[ R^*(x, u) = r_1 + \text{discounted rewards} \]
  along best trajectory starting in \( x_1 \)
- Optimal policy \( h^* \) can be computed from \( R^* \)
Algorithms

- Many algorithms available to find optimal $R^*, h^*$
- Some require prior knowledge about problem, or data:
- Others work with no prior knowledge, and collect data by online interaction:
Algorithms

- Many algorithms available to find optimal $R^*$, $h^*$
- Some require **prior knowledge** about problem, or data:

- Others work with no prior knowledge, and collect data by **online interaction**:
Algorithms

- Many algorithms available to find optimal $R^*$, $h^*$
- Some require prior knowledge about problem, or data:
- Others work with no prior knowledge, and collect data by online interaction:
1 Reinforcement learning

2 Challenge & contribution

3 Examples

4 Conclusions
Challenge

- Classical algorithms have to store $R(x, u)$ for every combination of state $x$ and action $u$
  - Only possible for when number of combinations is small
  - However, $x$ and $u$ often continuous
    - $\Rightarrow$ infinitely many combinations!
  - E.g., for robot, $x = [X, Y]$ continuous
Challenge

- Classical algorithms have to store $R(x, u)$ for every combination of state $x$ and action $u$

  - Only possible for when number of combinations is small
  - However, $x$ and $u$ often continuous
    \[ \Rightarrow \text{infinitely many combinations!} \]
  - E.g., for robot, $x = [X, Y]$ continuous
Challenge

- Classical algorithms have to store $R(x, u)$ for every combination of state $x$ and action $u$

- Only possible for when number of combinations is small
- However, $x$ and $u$ often continuous
  $\Rightarrow$ infinitely many combinations!
- E.g., for robot, $x = [X, Y]$ continuous
Challenge

- Classical algorithms have to store $R(x, u)$ for every combination of state $x$ and action $u$

- Only possible for when number of combinations is small

- However, $x$ and $u$ often continuous
  - $\Rightarrow$ infinitely many combinations!

- E.g., for robot, $x = [X, Y]$ continuous
Contribution

- Algorithms for reinforcement learning in problems with continuous states and actions
- Using approximate representations of returns
Example: RL for inverted pendulum

- **Goal**: point up and stay there
- **Difficulty**: insufficient power, need to *swing back & forth*
- **Reward**: the closer to vertical, the larger
Example: RL for robot goalkeeper

- Catch ball using only video camera image
Summary

- Reinforcement learning: very general framework
- Can learn from interaction, without prior knowledge

However:
- Classical RL algorithms do not work when states, actions continuous
- Need to use approximate representations (this thesis)
Reinforcement learning: very general framework
Can learn from interaction, without prior knowledge

However:
Classical RL algorithms do not work when states, actions continuous
Need to use approximate representations (this thesis)
Application areas

1. Control engineering: optimal & learning control (e.g., robot control)
2. Computer science: intelligent agents
3. Economics
4. etc.
Thank you!
Questions?