How to Train Your Robot Like a Dog

M. Mahdi Ghazaei Ardakani

Dept. of Automatic Control
LTH, Lund University

Friday Seminar
June 5th, 2015

Outline

- Introduction
- Reinforcement Learning (RL)
- LQR vs. RL
- Beyond Dynamic Programming
- Three-finger Hand
- Conclusion and Future research

Introduction

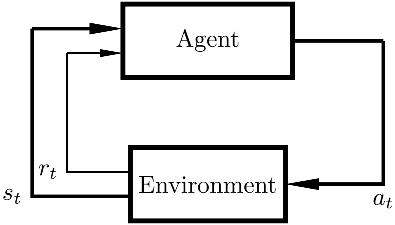
Behavior shaping by conditioning

1. Bring the dog into a desired position or action
2. Give immediate rewards
3. Be consistent

Introduction

The reinforcement Learning problem

- Agent
- World
- Reward



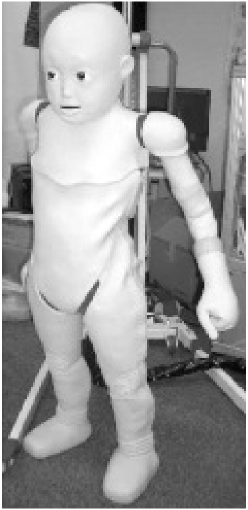
The objective is to maximize total expected discounted return

$$V_\pi(s_k) = E\left\{\sum_{i=k}^{\infty} \gamma^{i-k} r(s_i, a_i)\right\}, \qquad 0 < \gamma \leq 1. \tag{1}$$

Introduction

Principles of developmental robotics

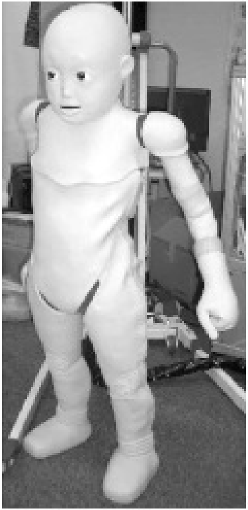
- **Incremental Developing**: continuous development and integration of new skills
- **Subjectivity**: what the robot learns must be a function of what the robot has experienced through its own sensors and effectors
- **Embodiment**
- **Grounding** How can the semantic interpretation of a formal symbol system be made intrinsic to the system
- **Verification**: an AI system can create and maintain knowledge only to the extent that it can verify that knowledge itself



Introduction

Principles of developmental robotics

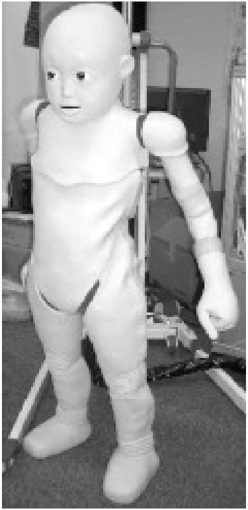
- **Incremental Developing**: continuous development and integration of new skills
- **Subjectivity**: what the robot learns must be a function of what the robot has experienced through its own sensors and effectors
- **Embodiment**
- **Grounding** How can the semantic interpretation of a formal symbol system be made intrinsic to the system
- **Verification**: an AI system can create and maintain knowledge only to the extent that it can verify that knowledge itself



Introduction

Principles of developmental robotics

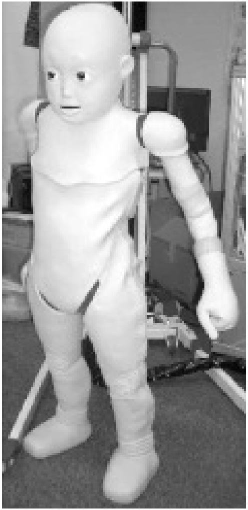
- **Incremental Developing**: continuous development and integration of new skills
- **Subjectivity**: what the robot learns must be a function of what the robot has experienced through its own sensors and effectors
- **Embodiment**
- **Grounding** How can the semantic interpretation of a formal symbol system be made intrinsic to the system
- **Verification**: an AI system can create and maintain knowledge only to the extent that it can verify that knowledge itself



Introduction

Principles of developmental robotics

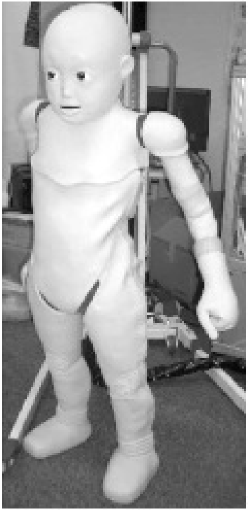
- **Incremental Developing**: continuous development and integration of new skills
- **Subjectivity**: what the robot learns must be a function of what the robot has experienced through its own sensors and effectors
- **Embodiment**
- **Grounding** How can the semantic interpretation of a formal symbol system be made intrinsic to the system
- **Verification**: an AI system can create and maintain knowledge only to the extent that it can verify that knowledge itself



Introduction

Principles of developmental robotics

- **Incremental Developing**: continuous development and integration of new skills
- **Subjectivity**: what the robot learns must be a function of what the robot has experienced through its own sensors and effectors
- **Embodiment**
- **Grounding** How can the semantic interpretation of a formal symbol system be made intrinsic to the system
- **Verification**: an AI system can create and maintain knowledge only to the extent that it can verify that knowledge itself



Outline

- Introduction
- **Reinforcement Learning**
- LQR vs. RL
- Beyond Dynamic Programming
- Three-finger Hand
- Conclusion and Future research

Reinforcement Learning

Bellman equation

$$V_\pi(s_k) = r(s_k, a_k) + \gamma V_\pi(s_{k+1}), V_\pi(0) = 0 \tag{2}$$

DT Hamiltonian

$$H(s_k, \pi(s_k), \Delta V_k) = r(s_k, \pi(s_k)) + \gamma V_\pi(s_{k+1}) - V_\pi(s_k) \tag{3}$$

From Bellman equation

$$H(s_k, \pi(s_k), \Delta V_k) = 0 \tag{4}$$

According to the principle of optimality, we derive discrete-time Hamilton-Jacobi-Bellman (HJB)

$$V^*(s_k) = \max_{\pi(.)} \left(r(s_k, a_k) + \gamma V^*(s_{k+1})\right) \tag{5}$$

And optimal policy

$$\pi^*(s_k) = \arg\max_{\pi(.)} \left(r(s_k, a_k) + \gamma V^*(s_{k+1})\right) \tag{6}$$

Reinforcement Learning

Approaches based on dynamic programming

- Policy Iteration

$$\pi_0 \xrightarrow{Eval} V^{\pi_0} \xrightarrow{Imp} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_*$$

- Value Iteration

$$V_{j+1}(s_k) = r(s_k, \pi_j(s_k)) + \gamma V_j(s_{k+1})$$

$$\pi_{j+1}(s_k) = \arg\max_{\pi(.)} \left(r(s_k, \pi(s_k)) + \gamma V_{j+1}(s_{k+1}) \right)$$

Outline

- Introduction
- Reinforcement Learning
- **LQR vs. RL**
- Beyond Dynamic Programming
- Three-finger Hand
- Conclusion and Future research

LQR vs. RL

$$s \to x, \quad a \to u, \quad \pi \to K, \quad r(x_k, u_k) = -(x_k^T Q x_k + u_k^T R u_k)$$

- Policy Iteration: Hewer's method for solving the DT Riccati equation

$$(A - BK_j)^T P_{j+1} (A - BK_j) - P_{j+1} + Q + K_j^T R K_j = 0$$

  and policy update

$$K_{j+1} = (R + B^T P_{j+1} B)^{-1} B^T P_{j+1} A$$

- Value Iteration studied by Lancaster and Rodman

$$P_{j+1} = (A - BK_j)^T P_j (A - BK_j) + Q + K_j^T R K_j$$

Outline

- Introduction
- Reinforcement Learning
- LQR vs. RL
- **Beyond Dynamic Programming**
- Three-finger Hand
- Conclusion and Future research

Reinforcement Learning

Dynamic programming

- Off-line procedure
- Full knowledge of A and B

Using data measured along the trajectory

- Adaptive Dynamic Programming (ADP)
- Neurodynamic programming (NDP)
- Actor-critic Architecture

Key ingredients

- temporal difference (TD)
- value function approximation (VFA)

RL can offer an (in)**direct adaptive control** approach

Reinforcement Learning

Dynamic programming

- Off-line procedure
- Full knowledge of A and B

Using data measured along the trajectory

- Adaptive Dynamic Programming (ADP)
- Neurodynamic programming (NDP)
- Actor-critic Architecture

Key ingredients

- temporal difference (TD)
- value function approximation (VFA)

RL can offer an **(in)direct adaptive control** approach

Q-Learning

Temporal Difference (TD) Error:

$$e_k = r + \gamma Q^i(x_{k+1}, \pi(x_{k+1})) - Q^i(x_k, u_k) \tag{7}$$

$$Q^{i+1}(x_k, u_k) = Q^i(x_k, u_k) + \eta e_k \tag{8}$$

The controller

$$\pi(x) = \arg\max_u Q(x, u) \tag{9}$$

Optimal value

$$V^*(x_k) = Q^*(x_k, \pi(x_k)) \tag{10}$$

ϵ-greedy policy: A random action with the probability of ϵ otherwise $u_k = \pi(x_k)$

Q Function for LQR

$$Q_K(x_k, u_k) = \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T \begin{bmatrix} Q + A^T P A & B^T P A \\ A^T P B & R + B^T P B \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} \equiv z_k^T H z_k$$

where P is the solution to Lyapunov equation for the given K

$$Q_K(x_k, u_k) = \bar{H}^T \bar{z}_k$$

where $\bar{H} = \mathrm{vec}(H)$ and $\bar{z}_k = z_k \otimes z_k$. This result in fixed-point equation

$$\bar{H}^T \bar{z}_k = x_k^T Q x_k + u_k^T R u_k + \bar{H}^T \bar{z}_{k+1}$$

by setting $\frac{\partial}{\partial u} Q_K(x_k, u) = 0$

$$u_k = -K x_k = -(H_{uu})^{-1} H_{ux} x_k \tag{11}$$

Outline

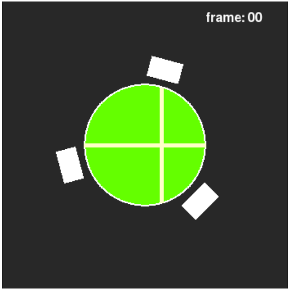
- Introduction
- Reinforcement Learning
- LQR vs. RL
- Beyond Dynamic Programming
- **Three-finger Hand**
- Conclusions

Three-finger Hand

Problem

Find an optimal sequence for fingers in order to rotate a ball counterclockwise at fast as possible

- Camera detects the rotation and drops
- Tactile sensor provides information for bad/good grips
- Internal states are observable

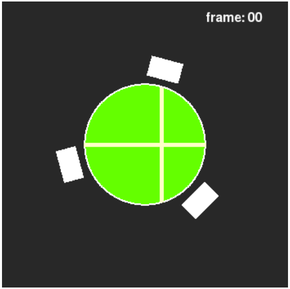


Three-finger Hand

Problem

Find an optimal sequence for fingers in order to rotate a ball counterclockwise at fast as possible

- Design a reward!



Three-finger Hand: Methods

Abstract States:

Left Open = $\{(r, \phi) | r > r_0 \wedge 0 < \phi - \phi_0 \leq 15\pi/180\}$

u_i	Effect
0	Null
1	Close or Open
2	Left or Right

Possible actions $3^3 = 27$

x_i	Definition
0	Right Open
1	Right Close
2	Left Open
3	Left Close

No. of states $4^3 = 64$

Three-finger Hand: Methods

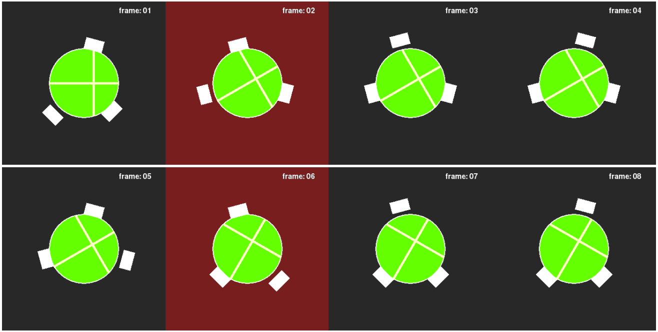
$$x(k+1) = f(x(k), u(k)) \tag{12}$$

where

$$\begin{aligned} f_i = &\operatorname{shl}(\operatorname{XOR}(\operatorname{shr}(x_i \wedge 10\mathsf{b}), \operatorname{shr}(u_i \wedge 10\mathsf{b}))) \\ &\wedge \operatorname{XOR}(x_i \wedge 01\mathsf{b}, u_i \wedge 01\mathsf{b}) \end{aligned} \tag{13}$$

$$r(x,u) = \begin{cases} 1 & \text{ccw rotation} \\ -1 & \text{cw rotation} \\ -1 & \text{one or two fingers are still while the rest are moving} \\ -2 & \text{all fingers move but not all in the same direction} \\ -10 & \text{unstable grip, i.e., less than 2 fingers in contact} \end{cases}$$

Three-finger Hand: Results

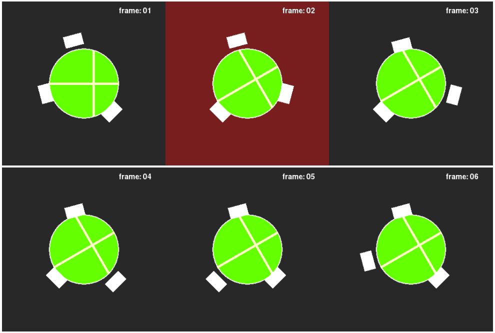


The optimal sequence $2/8$. The states right after a rotation are highlighted in red.

Three-finger Hand: Results

k	x_k^T	u_k^T	x_{k+1}^T	r_{k+1}
0	$[0,0,0]$	$[1,1,2]$	$\rightarrow [1,1,2]$	0
1	$[1,1,2]$	$[2,2,2]$	$\rightarrow [3,3,0]$	1
2	$[3,3,0]$	$[1,0,1]$	$\rightarrow [2,3,1]$	0
3	$[2,3,1]$	$[2,0,0]$	$\rightarrow [0,3,1]$	0
4	$[0,3,1]$	$[1,1,0]$	$\rightarrow [1,2,1]$	0
5	$[1,2,1]$	$[2,2,2]$	$\rightarrow [3,0,3]$	1
6	$[3,0,3]$	$[1,1,0]$	$\rightarrow [2,1,3]$	0
7	$[2,1,3]$	$[2,0,0]$	$\rightarrow [0,1,3]$	0
8	$[0,1,3]$	$[1,0,1]$	$\rightarrow [1,1,2]$	0

Three-finger Hand: Results



A non-optimal sequence $1/6$: one steps out of six cause a ccw rotation.

Three-finger Hand: Results

k	x_k^T	u_k^T	x_{k+1}^T	r_{k+1}
0	$[0, 0, 0]$	$[2, 1, 1]$	$\rightarrow [2, 1, 1]$	0
1	$[2, 1, 1]$	$[0, 2, 2]$	$\rightarrow [2, 3, 3]$	1
2	$[2, 3, 3]$	$[1, 0, 1]$	$\rightarrow [3, 3, 2]$	0
3	$[3, 3, 2]$	$[0, 0, 2]$	$\rightarrow [3, 3, 0]$	0
4	$[3, 3, 0]$	$[0, 1, 1]$	$\rightarrow [3, 2, 1]$	0
5	$[3, 2, 1]$	$[0, 2, 0]$	$\rightarrow [3, 0, 1]$	0
6	$[3, 0, 1]$	$[1, 1, 0]$	$\rightarrow [2, 1, 1]$	0

Three-finger Hand

- Incremental Developing
- Subjectivity
- Embodiment
- Grounding
- Verification

Three-finger Hand

- Incremental Developing
- Subjectivity
- Embodiment
- Grounding
- Verification

- Model free
- Rewards are entirely related to the objective
- The reward signal is grounded in the physical world

Three-finger Hand

- Incremental Developing
- Subjectivity
- Embodiment
- Grounding
- Verification

I get a kick out of rotating a ball counterclockwise
and it is so boring to drop it!

What is real?

Outline

- Introduction
- Reinforcement Learning
- LQR vs. RL
- Beyond Dynamic Programming
- Three-finger Hand
- **Conclusion and Future research**

Conclusion and Future research

- Reinforcement Learning
- Connection between RL and LQR
- Importance of verification principle

- Extension to 3D and continuous states
- How to reuse a learned model
- Automatic creation of abstract states
- Reward design and the role of intuition

Conclusion and Future research

- Reinforcement Learning
- Connection between RL and LQR
- Importance of verification principle

- Extension to 3D and continuous states
- How to reuse a learned model
- Automatic creation of abstract states
- Reward design and the role of intuition

Thank you for listening!

References

1. Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, Cambridge, MA, 1998
2. Frank L Lewis and Draguna Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9(3):32–50, 2009.
3. Alexander Stoytchev. Some basic principles of developmental robotics. *IEEE Tran. Autonomous Mental Development*, 1(2):122–130, 2009.
4. Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.
5. G Hewer, An iterative technique for the computation of the steady state gains for the discrete optimal regulator, *IEEE Tran. Automatic Control*, 16(4):382–384, 1971.
6. Peter Lancaster, Leiba Rodman. *Algebraic Riccati Equations*, Oxford Univ Press, London, U.K. 1995