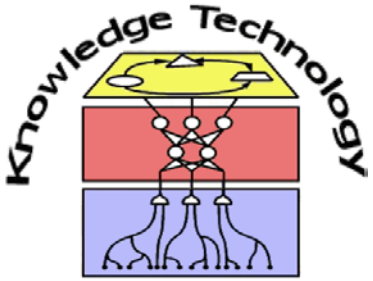
Real-World Reinforcement Learning for Autonomous Humanoid Robot Charging in a Home Environment

Nicolás Navarro, Cornelius Weber, Stefan Wermter
University of Hamburg
Dept. of Informatics, Knowledge Technology



http://www.informatik.uni-hamburg.de/WTM/

August 31, 2011

Outline

- Motivation
- Reinforcement learning (RL)
- Neural implementation
- SARSA algorithm
- Experimental results
- Conclusion

Motivation

- Need for studying humanoid robots within home environments
- Limited energetic capabilities of the Nao robot

http://ksera.ieis.tue.nl/

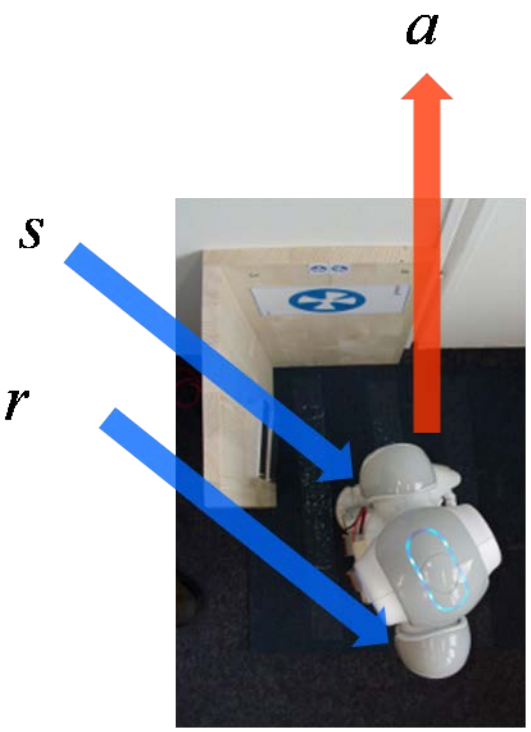
RobotDoC

Robotics for Development of Cognition

http://robotdoc.org/

Reinforcement learning

- Perceive state ***s***
- Perform action ***a***
- Occasionally, receive reward ***r***
- Perceive state ***s'***
- Perform action ***a'***



Markov Decision Process (MDP)

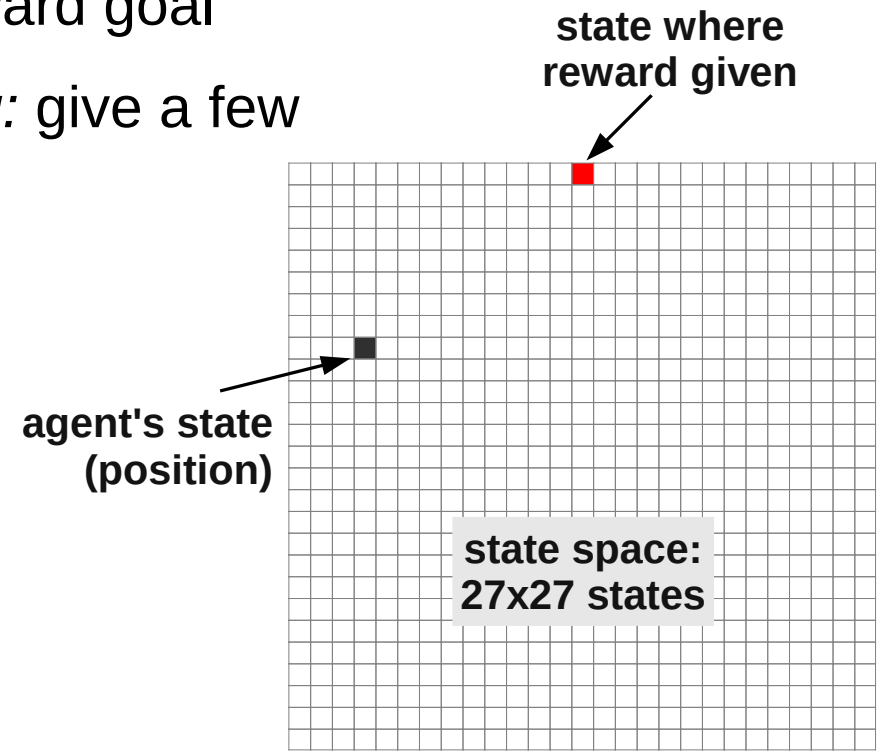
- Fixed transition probabilities
- Next move not depending on history
- Fixed reward probability

Concepts of SARSA and supervised RL

Objective: get to the reward quickly

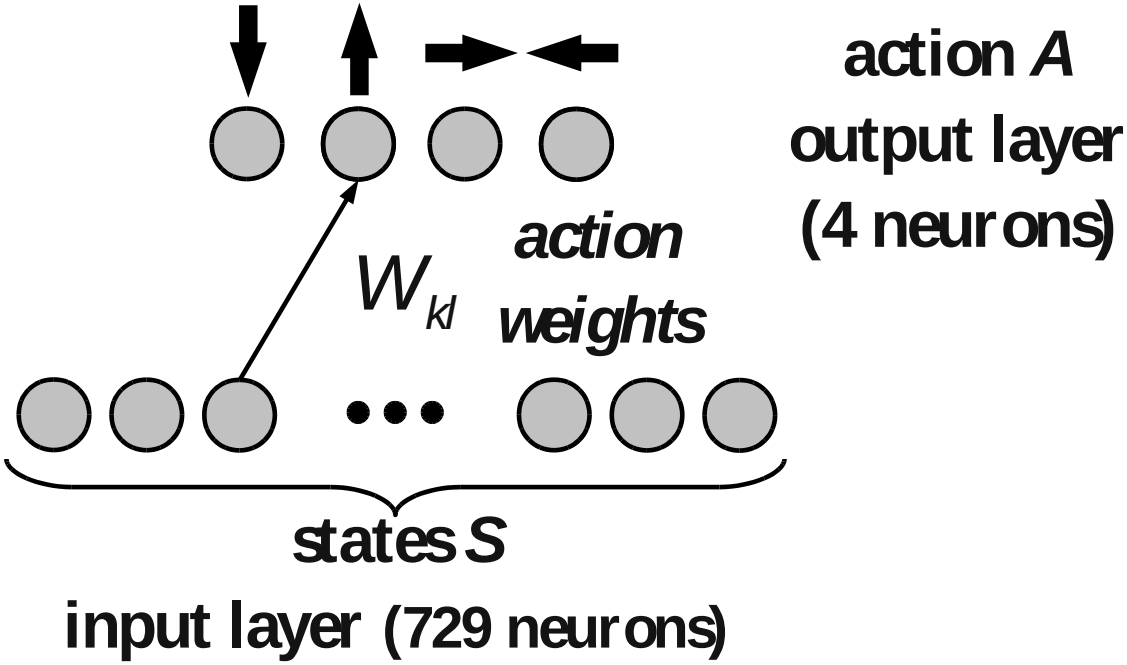
Principle: sample ***s***, ***a*** and ***r***, ***s***, ***a*** (SARSA) in MDP trials, and learn principle: state-action values Q(*s*, *a*), which increase toward goal

Supervised reinforcement learning: give a few correct training examples initially



Neural implementation

1-layer feed forward network maps state to action



SARSA algorithm

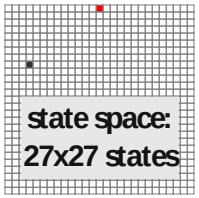
(i0)	Place the robot in a random position for a new trial	
(i1)	Compute states	s
(i2)	Compute action strength	$h_i = \sum_l W_{il} s_l$
(i3)	Select action	$P_{(a_i=1)} = \frac{e^{\beta h_i}}{\sum_k e^{\beta h_k}}$
(i4)	Current estimate	$Q_{(s,a)} = \sum_{k,l} W_{kl} a_k s_l$
Repeat until trial ends successfully (r=1)		
(0)	Execute action	a
(1)	Compute new states and check for reward	s'
(2)	Compute action strength	$h_i = \sum_l W_{il} s'_l$
(3)	Select action	$P_{(a'_i=1)} = \frac{e^{\beta h_i}}{\sum_k e^{\beta h_k}}$
(4)	Current estimate	$Q_{(s',a')} = \sum_{k,l} W_{k,l} a'_k s'_l$
(5)	Compute Prediction error	$\delta = (1-r)\gamma Q_{(s',a')} + r - Q_{(s,a)}$
(6)	Weight update	$\Delta W_{ij} = \epsilon \delta a_i s_j$
(7)	New turns old	$s, a \leftarrow s', a'$

SARSA algorithm

Gaussian state activation to “blur” the activation around the “currently active state unit”.

$$s_j = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{(x_j - \mu_x)^2 + (y_j - \mu_y)^2}{2\sigma^2}}$$

(i0)	Place the robot in a random position for a new trial	
(i1)	Compute state	s
(i2)	Compute action strength	$h_i = \sum_l W_{il} s_l$
(i3)	Select action	$P_{(a_i=1)} = \frac{e^{\beta h_i}}{\sum_k e^{\beta h_k}}$
(i4)	Current estimate	$Q_{(s,a)} = \sum_{k,l} W_{kl} a_k s_l$
(0)	Execute action	a
(1)	Compute new states and check for reward	s'
(2)	Compute action strength	$h_i = \sum_l W_{il} s'_l$
(3)	Select action	$P_{(a'_i=1)} = \frac{e^{\beta h_i}}{\sum_k e^{\beta h_k}}$
(4)	Current estimate	$Q_{(s',a')} = \sum_{k,l} W_{k,l} a'_k s'_l$
(5)	Compute Prediction error	$\delta = (1-r)\gamma Q_{(s',a')} + r - Q_{(s,a)}$
(6)	Weight update	$\Delta W_{ij} = \epsilon \delta a_i s_j$
(7)	New turns old	$s, a \leftarrow s', a'$



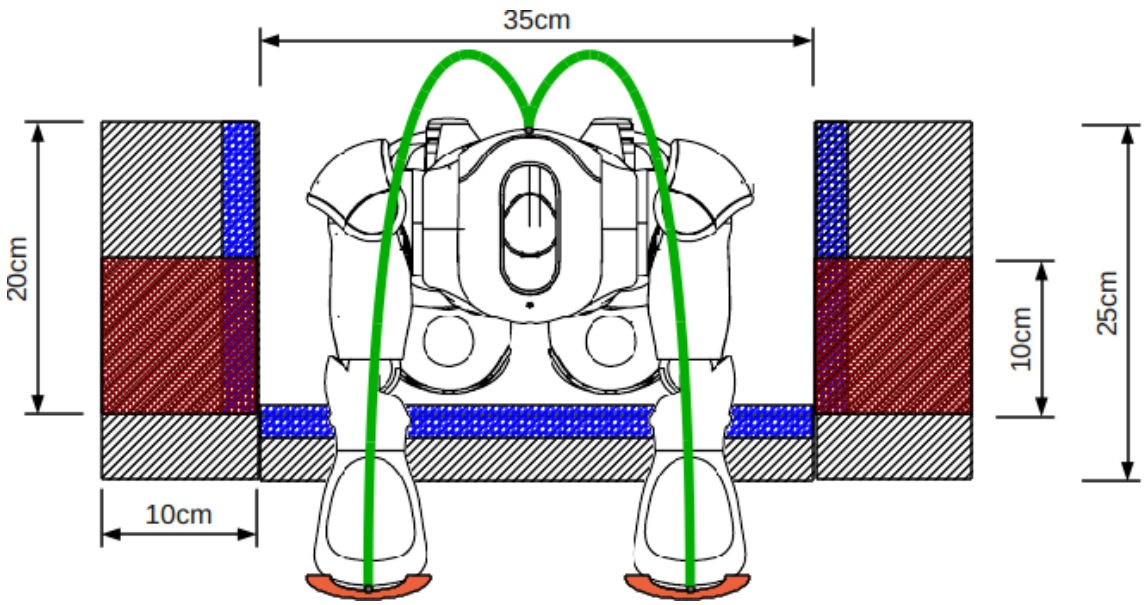
Analysis of Supervised Reinforcement Learning

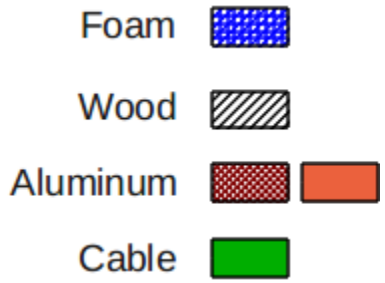
Table summarizes avg. # of steps to solve a trial after training (taken over 10 trials)

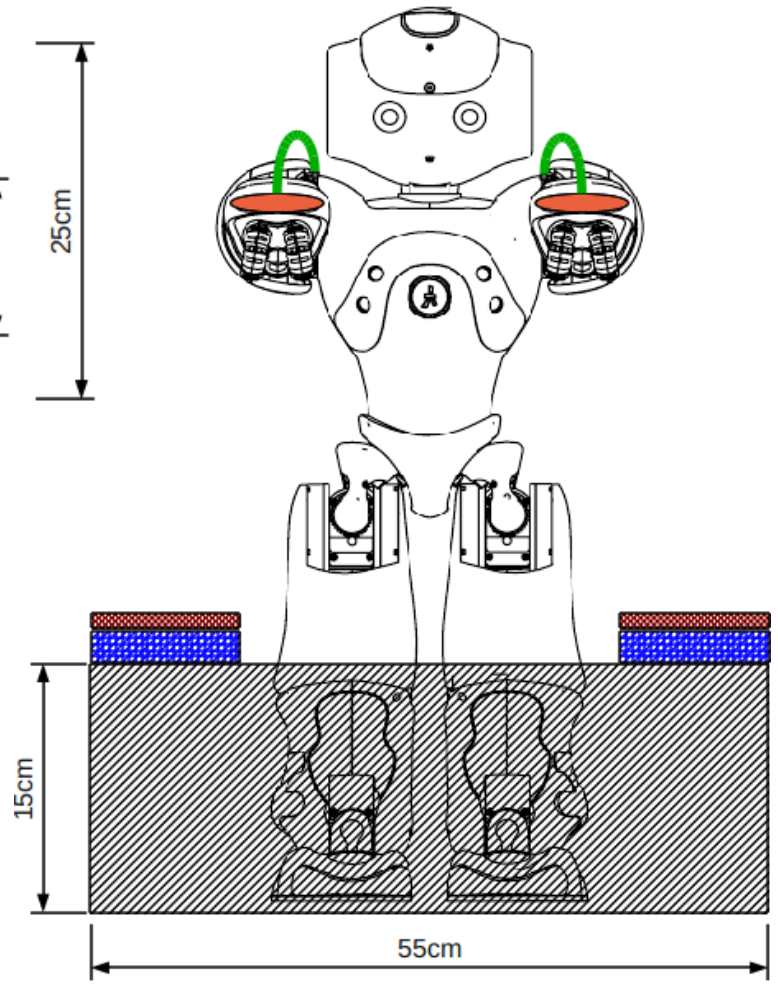
Tele-operated by User				
state activation	Single		**Gaussian***	
off-line training trials	180	300	180	300
avg. # of steps	111.90	86.10	56.10	**39.60**

- 10 training examples generated by user with avg. # of steps = 39.6
- Avg # of steps = 451.26 during random exploration

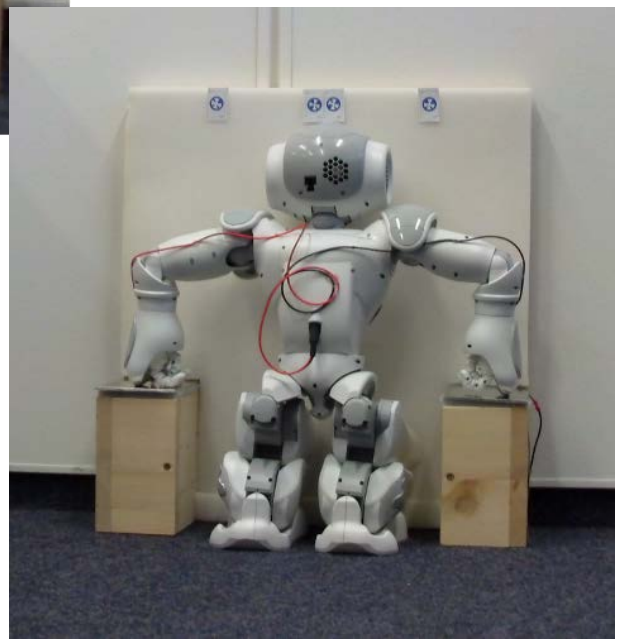
Towards a solution: First prototype



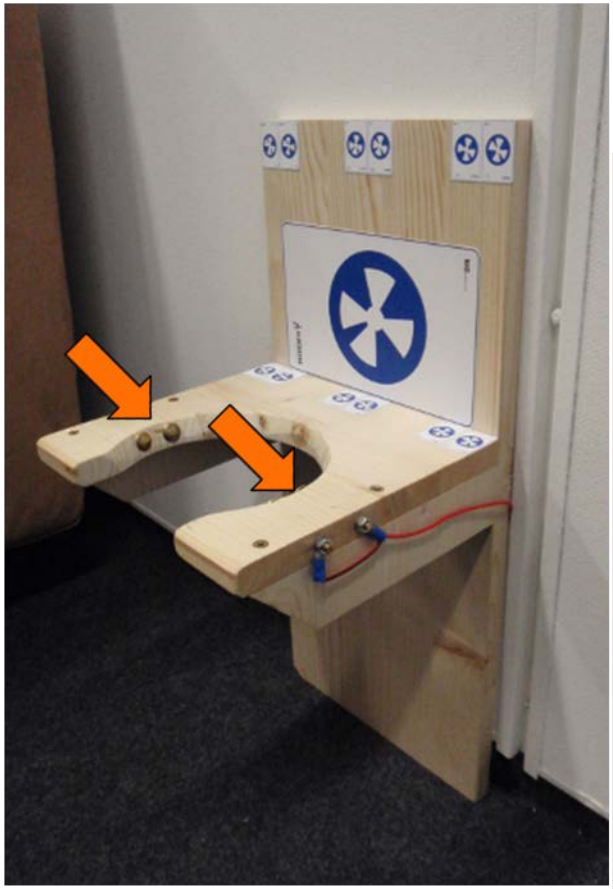


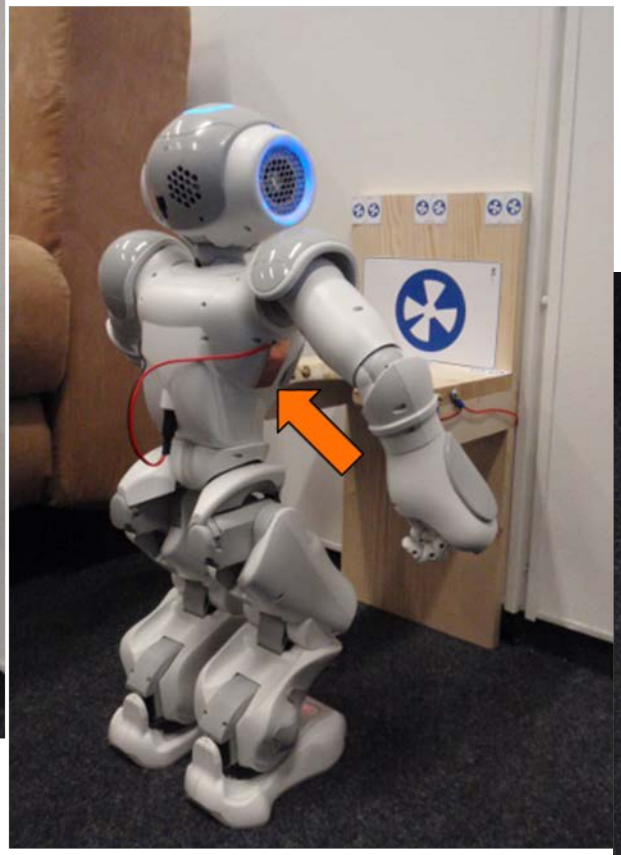


Towards a solution : First prototype



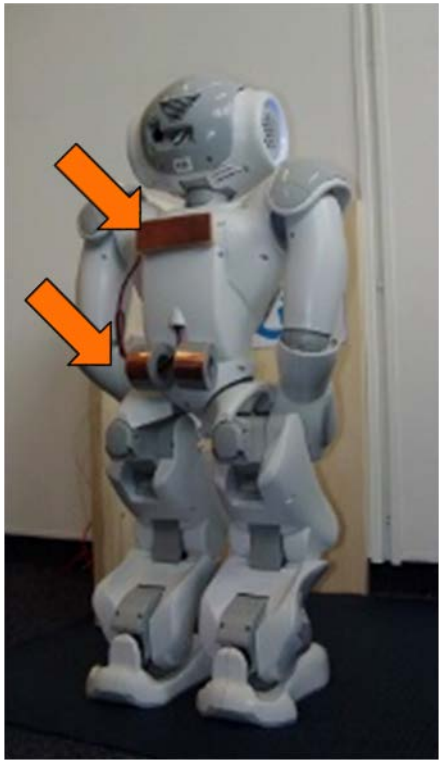
Towards a solution : Second prototype





Aluminum

Towards a solution : Third prototype



Aluminum

Landmark

Towards a solution: Robot behaviour

Towards a solution: Robot behaviour

- Phase I

Hard coded: search and approach landmarks. Places the robot 40 cm away from landmarks

- Phase II

Hard coded: Places the robot (approx.) parallel to the wall looking at the landmarks

- Phase III

Neural Docking: “SARSA”. After learning the robot senses position and orientation and manoeuvres towards goal

- Phase IV

Hard coded: check sensors. If false positive is detected, correct pose or go to Phase III. Else move the robot to a crouch pose

Experimental results

Table summarizes no steps needed to solve 10 trials after training

State activation	Action-state pairs learned (%)	# of success	# false positive	# aborted	Avg. # steps on success	Std. Deviation
Single	4	**6**	**1**	**3**	23,80	8,23
Gaussian*	34	**5**	**3**	**2**	23,60	14,30

- Grid world of 11x11x15 = 1,815 states
- 6 actions: forward, backward, m_right, m_left, turn_right, turn_left
- 50 training examples generated by user
- Off-line training: 300 trials

Demonstration

Conclusion

- Use of appropriate training examples (“supervised”) proved to be a key factor for real-world learning scenarios.
- Gaussian distributed states activation has a helpful state space reduction effect.

Thank you for your attention!

Any questions?

Acknowledgments: This research has been partly supported by the **EU project RobotDoC** under 235065 ROBOT-DOC from the 7th Framework Programme, Marie Curie Action ITN and by the **KSERA project** funded by the European Commission under the 7th Framework Programme (FP7) for Research and Technological Development under grant agreement n◦ 2010-248085.