



# LeCoPro

Learning Control of Production Machines



Vrije Universiteit Brussel

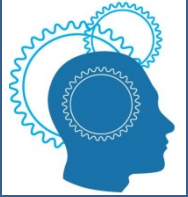


## Reinforcement learning for repetitive systems with discrete sensor information

S. Goossens, G. Pinte, W. Symens  
FMTC, Leuven

M. Gagliolo, A. Rodriguez, A. Nowé  
CoMo, VUB, Brussels

Speaker: Matteo Gagliolo (VUB)



# Control tasks with discrete sensors

“Classic” control:

- Continuous measurement output signal
- Calculations based on continuous signal

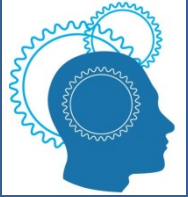
**Issues:**

- Continuous measurement not always possible

*Ex. Position measurement of a **moving object***

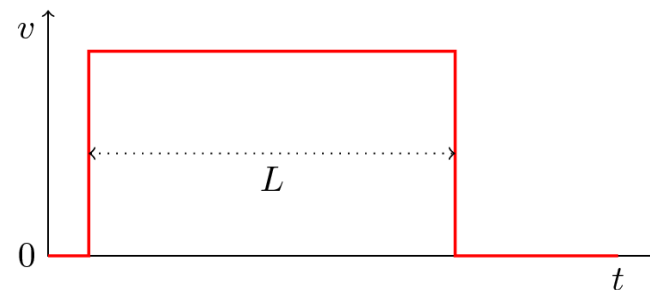
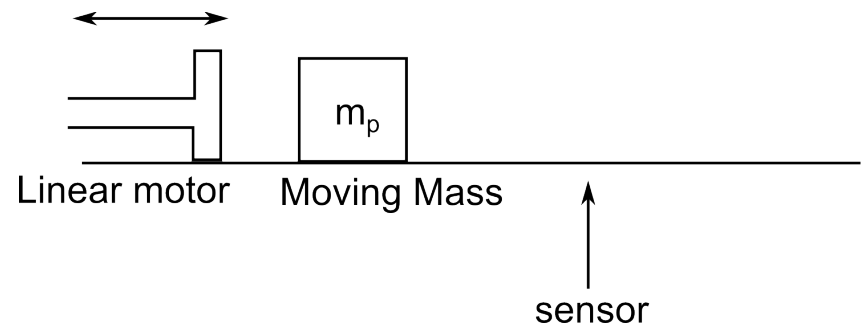
- Measuring accurate over a long stroke is difficult/expensive

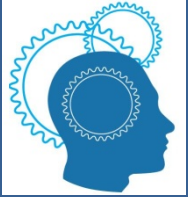
*Ex. Measure only available at **discrete locations***



# Position control tasks (FMTC)

- A linear motor “punches” a cart along rails
- Control signal: speed, one parameter (length  $L$  mapped to  $[0,1]$ )
- **Discrete** sensor along the rails
- **Task A**: pass the sensor at a given time
- **Task B**: stop in front of the sensor





# Machine Learning for control

- **Supervised** learning: a target output is available. Example: tracking a signal

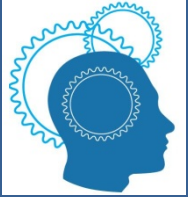
*Given input  $\mathbf{s}(t)$ , the correct answer is  $\mathbf{u}^*(t)$ .*

*Examples: Wavelet analysis, Recurrent neural networks.*

- **Semi-supervised** learning: no target output, but some performance measure is available.

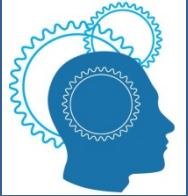
*Given input  $\mathbf{s}(t)$ , you answer  $\mathbf{u}(t)$ , and receive a “score”,  
telling you *how good*  $\mathbf{u}(t)$  is.*

*Examples: Genetic Algorithms: fitness function  $g(\mathbf{u})$ . Reinforcement Learning: reward function  $r(t)$ .*



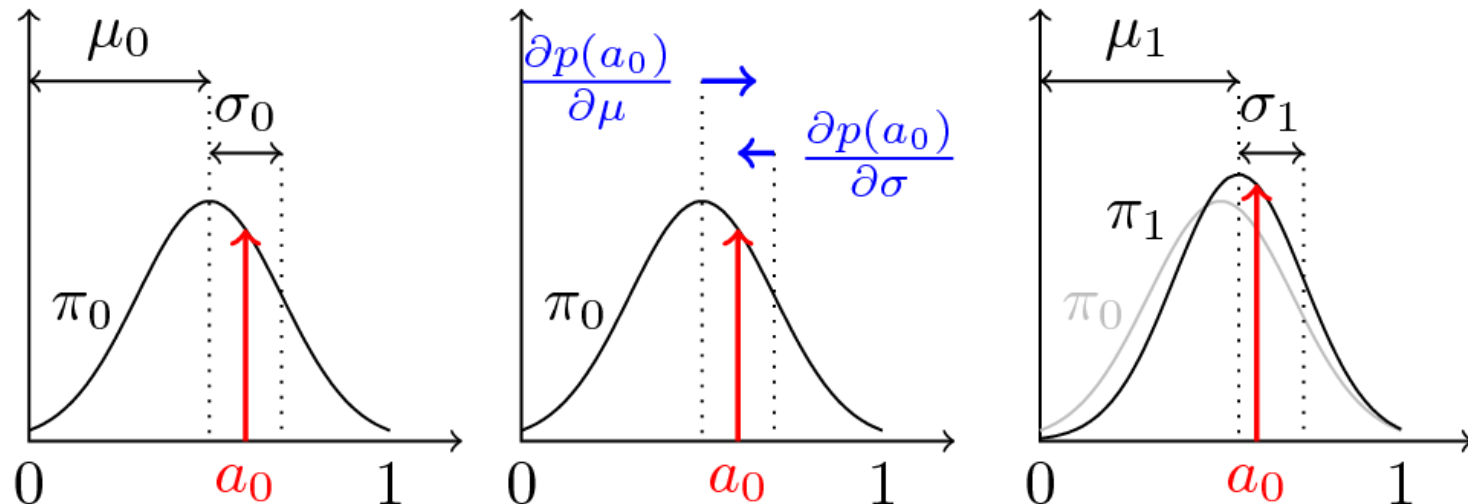
# Reinforcement Learning (RL)

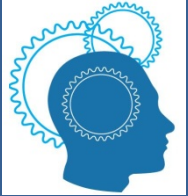
- Basic idea: actions are chosen by a stochastic *policy*. Actions yielding a high reward are *reinforced*, becoming more likely.
- Two RL algorithms were tested on both tasks:
  - **Policy gradient**: parametric policy, parameters are updated following an estimate of the gradient of the reward. Starting from a candidate solution, this method converges to the closest local optimum.
  - **Learning Automata**: non-parametric policy, updated reinforcing actions based on their reward. Starts from a uniform distribution over actions.



# Policy gradient

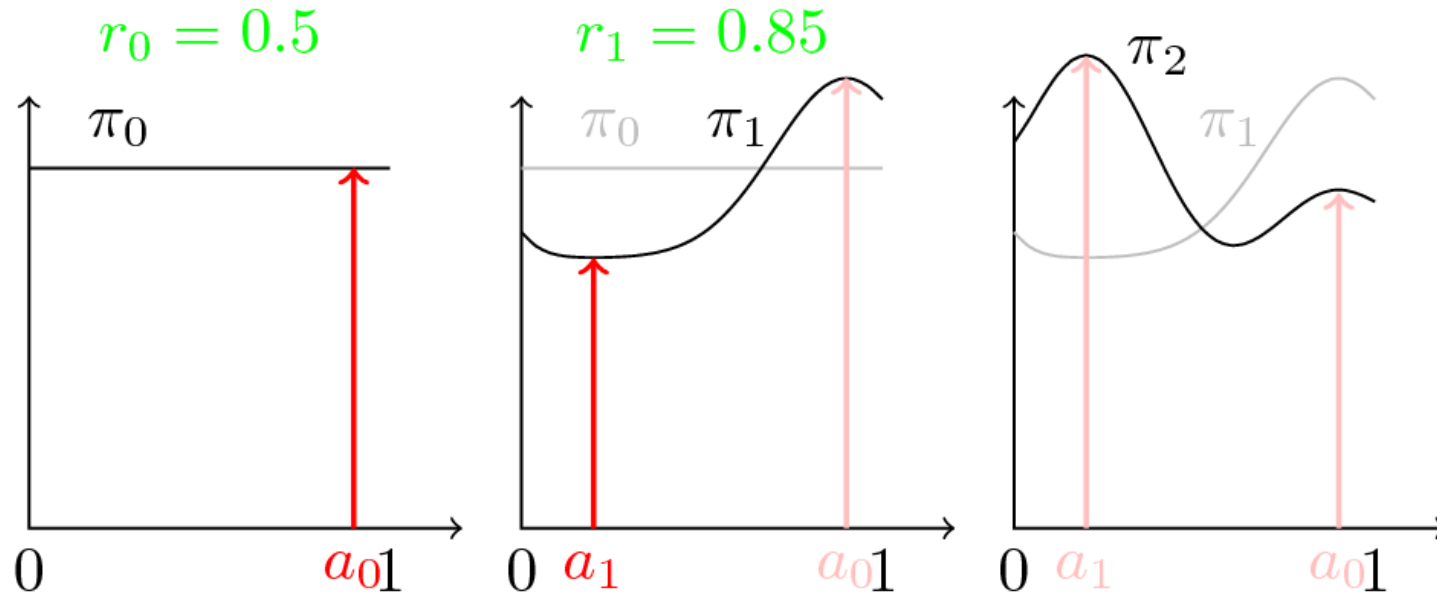
**Parametric** pdf over actions (e.g. Gaussian). Parameters are updated following an **estimated gradient** of the expected reward.

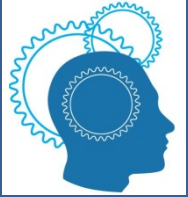




# Learning Automata

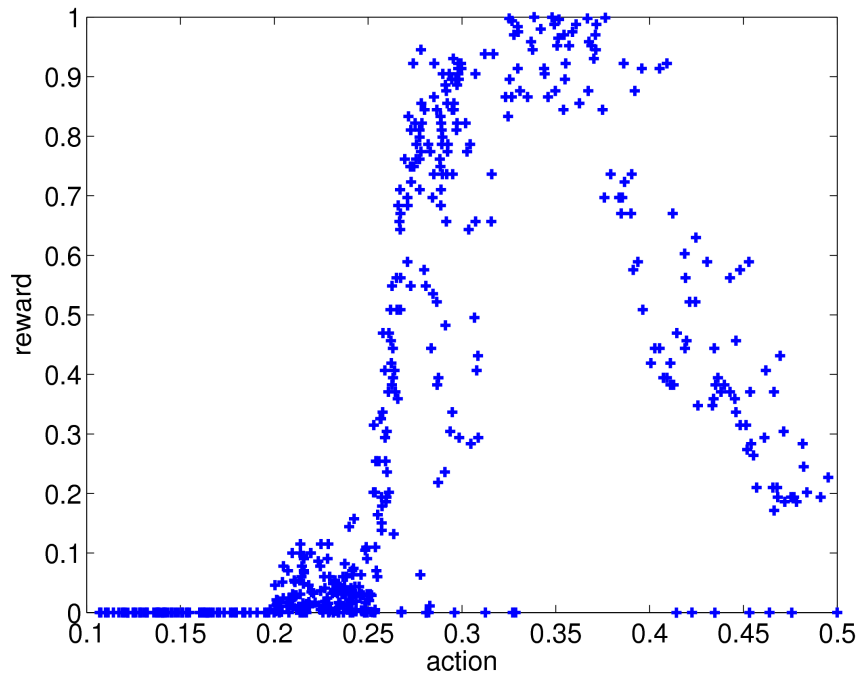
**Nonparametric** pdf over actions (mixture of Gaussians): when an action is taken, its probability is increased proportionally to the reward observed.



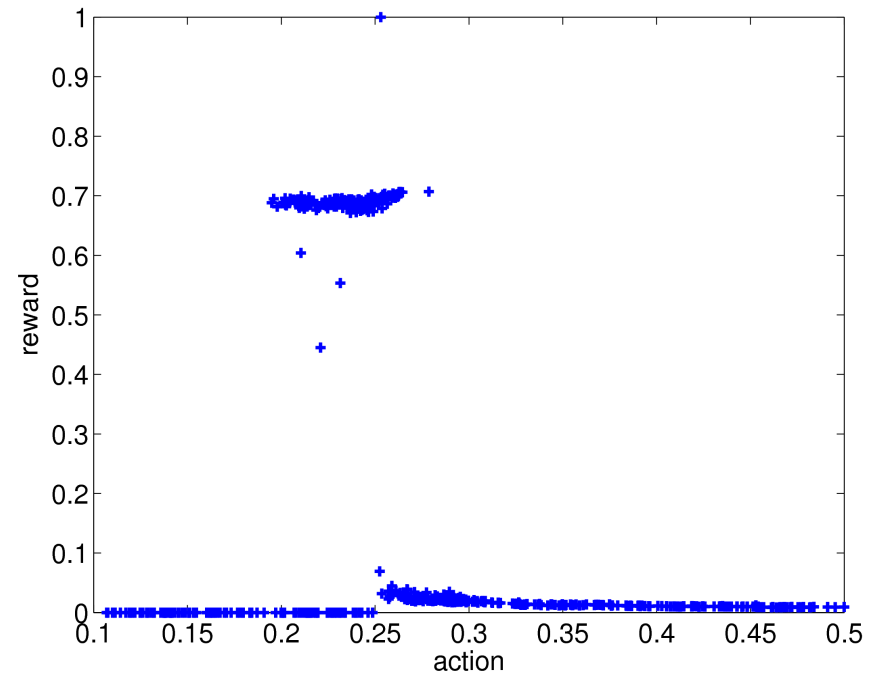


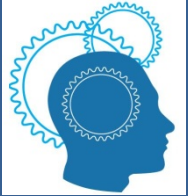
# Reward functions

Task A (time)



Task B (position)

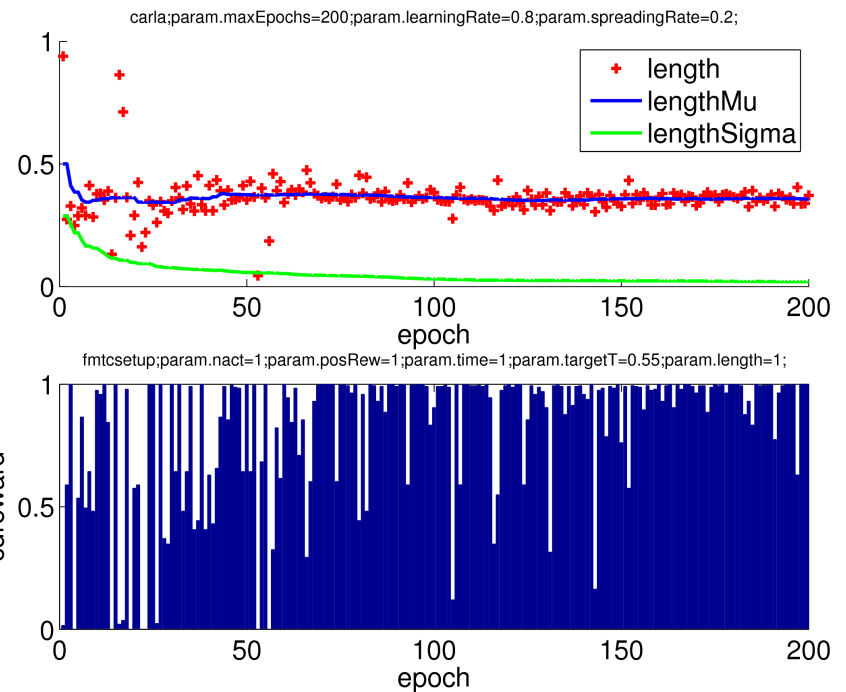
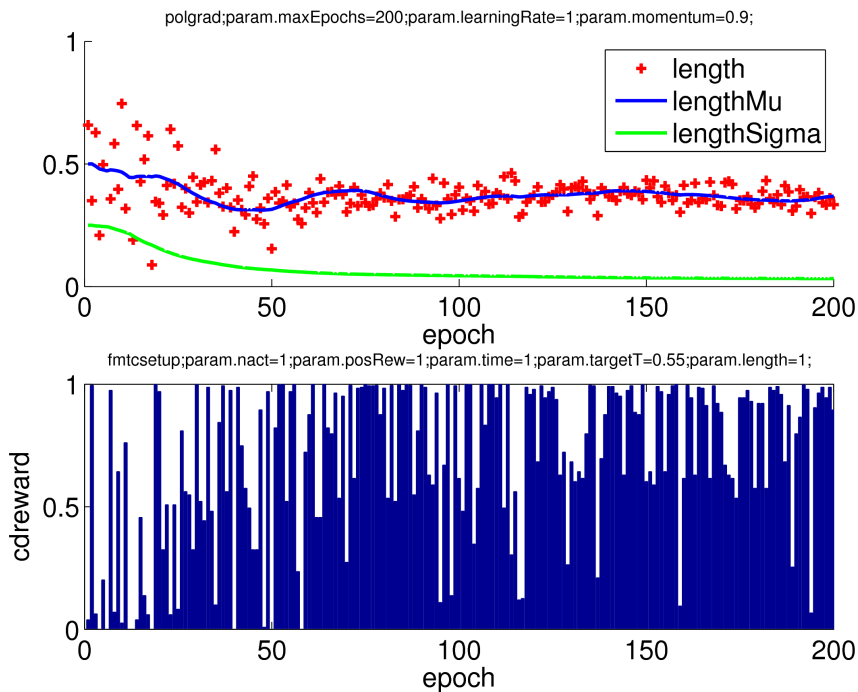


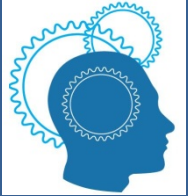


# Task A: time control

## Policy Gradient

## Learning Automata

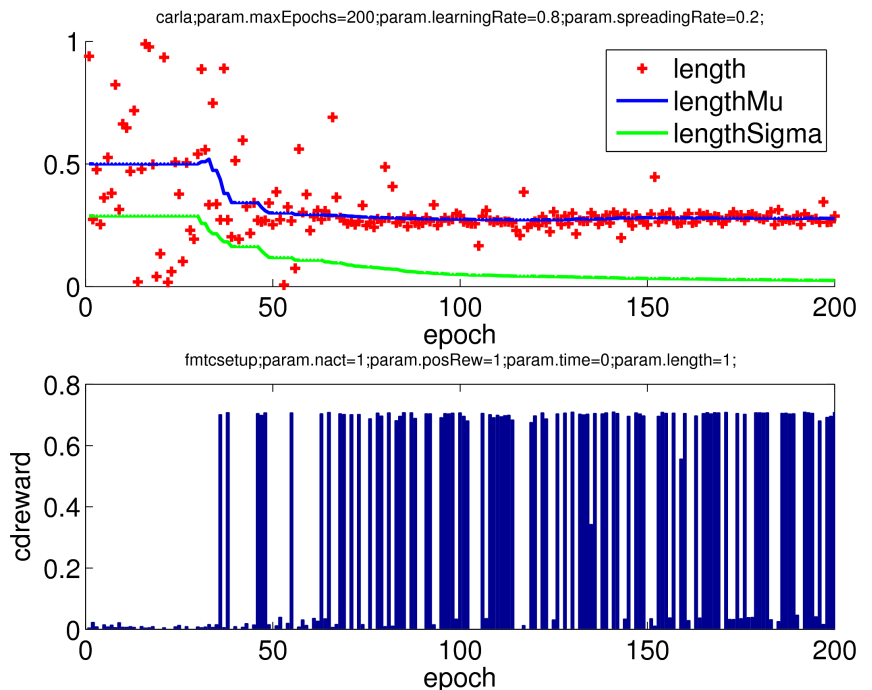
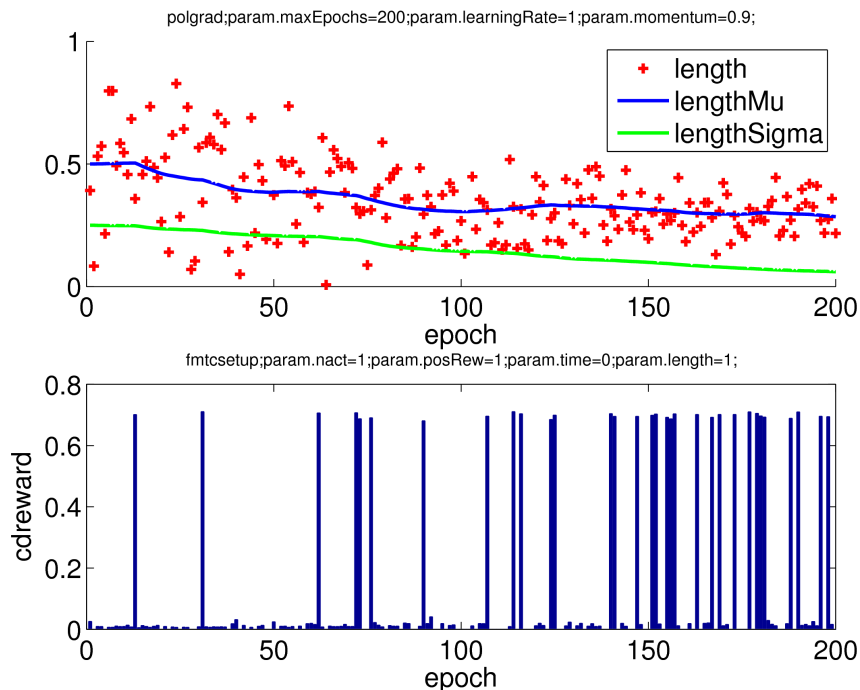


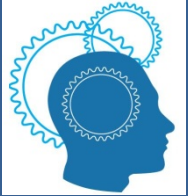


# Task B: position control

## Policy Gradient

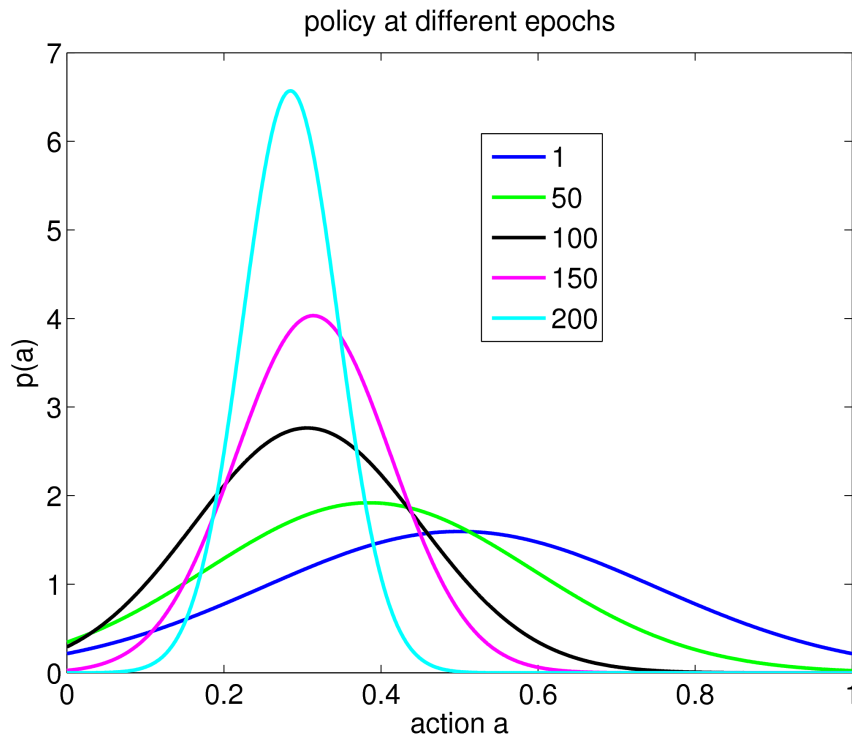
## Learning Automata



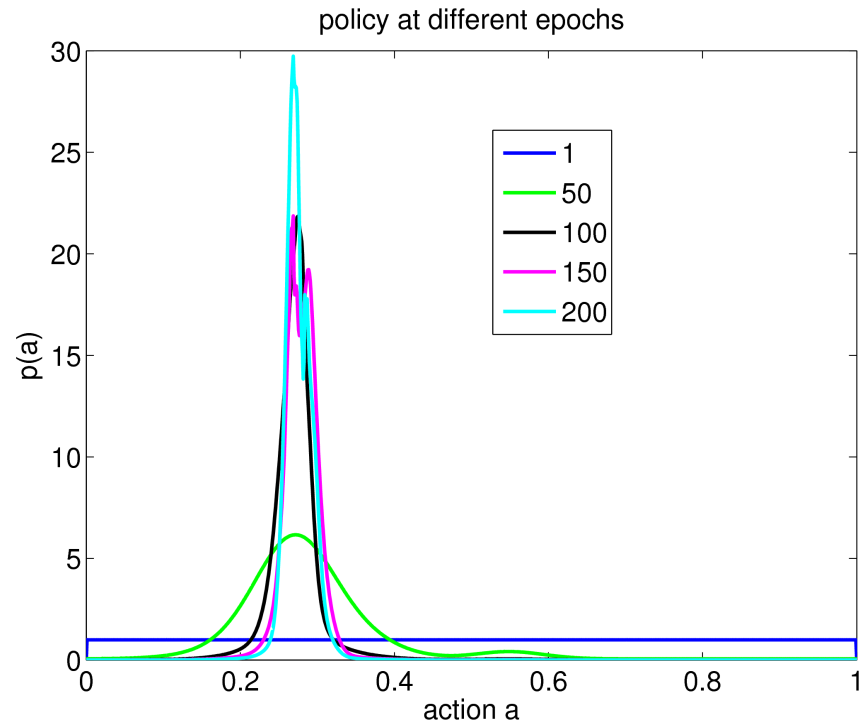


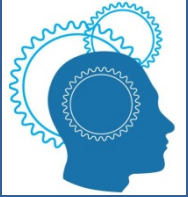
# Task B: position control

## Policy Gradient



## Learning Automata





# Conclusions and ongoing work

## Conclusions

- Task B is much harder (a “needle in a haystack”)
- Both methods can solve both tasks, but LA is better on task B

## Ongoing work

- Use LA as a preliminary search for PG
- Control multiple parameters (*length, amplitude*)
- Multistage LA
- Test Parameter Exploring PG [Sehnke et al. 2010]
- Experiments on the wetclutch

## References

- [1] Peters, Schaal, “Policy gradient methods for robotics”, *IEEE Conf. on Intelligent Robots and Systems*, 2006
- [2] Rodriguez, Grau, Nowé, “Continuous Actions Reinforcement Learning Automata. Performance and Convergence”, *Intl. Conf. on Agents and Artificial Intelligence (ICAART)*, 2011