

Reinforcement learning in robotics, an application using Lego Mindstorms

Special project for Robotics course
Prof. Giuseppina Gini
Politecnico di Milano

Marco Cioffi (Matr. 675053)

1

All content of this work (including text and original images), unless otherwise noted, is licensed under the Creative Commons Attribution NonCommercial license. For other uses please contact the author at mcioffi_AT_gmail.com. You can read the full text of the license at <http://creativecommons.org/licenses/by-nc/2.5/>

Outline

- Introduction
- Reinforcement learning
 - Principal ideas
 - General model
 - Types of reinforcement learning
 - Q-Learning
- LEGO Mindstorms
- Q-Learning and LEGO Mindstorms
 - Robot Tony
 - Implementation
- Conclusions

2

Introduction

□ Machine Learning

- The area of artificial intelligence concerned with the development of techniques which **allow computers to "learn"**. More specifically, machine learning is a method for creating computer programs by the analysis of data sets. Machine learning overlaps heavily with **statistics**, since both fields study the analysis of data, but unlike statistics, machine learning is concerned with the algorithmic complexity of computational implementations.

[wikipedia.org]

3

The types of learning

- Machine Learning for "The Journal of Machine Learning", Springer
- Supervised and unsupervised learning methods
 - learning decision and regression trees, rules, connectionist networks, probabilistic networks and other statistical models, inductive logic programming, case-based methods, ensemble methods, clustering, etc...
- **Reinforcement learning;**
- Evolution-based methods;
- Explanation-based learning;
- Analogical learning methods;
- Automated knowledge acquisition;
- Learning from instruction;
- Visualization of patterns in data;
- Learning in integrated architectures;
- Multistrategy learning;
- Multi-agent learning;

4

Approaches on learning

- Supervised learning
 - Human tells to AI which is the result of an experiment (e.g. shape of an object, input: a picture, output: the shape of the object).
 - Training data: (X,Y). (features, label)
 - Regression, Classification
- Unsupervised learning
 - The AI decides autonomously, without human intervention, which is the result of an experiment
 - Training data: X. (features only)
 - Clustering

5

Reinforcement Learning

- *“Reinforcement learning (RL) is learning from **interaction** with an **environment**, from the **consequences of action**, rather than from explicit teaching. RL become popular in the 1990s within machine learning and artificial intelligence, but also within operations research and with offshoots in psychology and neuroscience.”*

Reinforcement Learning: A Survey
(Kaelbling, Moore)

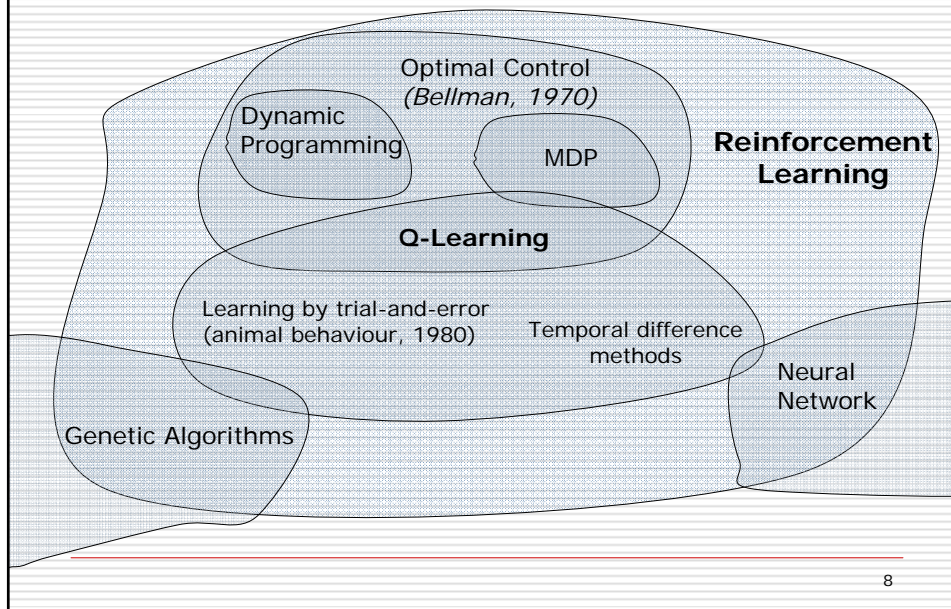
6

Reinforcement Learning – roots

- Three fundamental roots
 - Psychology roots in **animal learning**
(Trial and Error, Behaviorism → Pavlov, Thorndike, Skinner)
 - Issue of **optimal control** and its solution using dynamic programming
(Bellman, ~1960)
 - Ideas of learning based on **temporal difference** (Samuel, ~ 1960)
- Central is the Q-Learning (Watkins, 1989)

7

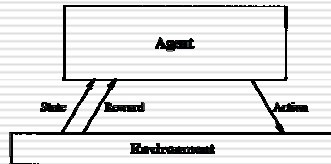
A whole view



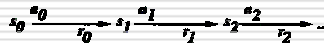
8

General model (1)

Concepts:
State (S)
Action (A)
Reward (R)



Learning from interaction



- The agent has to learn a control logic (sequence of decision) which maximize the sum of rewards.

$\pi : S \rightarrow A$

(control policy)

9

General model (2)

- Making a relation between states and actions it is possible to **numerically maximize the reward function** during the time.
- This kind of learning, compared to other models, is classifiable as a multi-level process (**Markovian**).
- An agent that use Reinforcement Learning must use a **trial-and-error** process (this is not completely supervised but interactive)
- The actions could change due to both immediate rewards and future rewards (**Delayed effect**).
- Elements:
 - Policy (state space \rightarrow action space)
 - Reward function (each state \rightarrow real number)
 - Value function (each state \rightarrow total expected reward)

10

Markov decision process (1)

- **Andrei Markov (1913)**
- The conditioned probability of the next state depends only from the history of precedent states.
 - k -th order Markov Process
- Formally
 - Initial State: S_0
 - Reward Function: $R(s)$
 - Transition Model:
 - $T(s,a,s')$ (deterministic)
 - $T(s,a,P(s'))$ (stochastic)

11

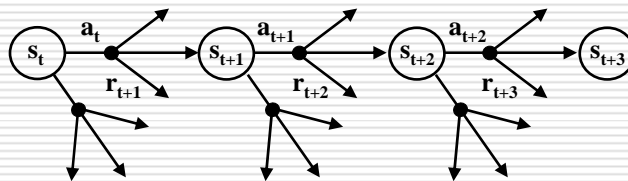
Markov decision process (2)

- Random variable:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k}$$

- State's attended value using a specified policy:

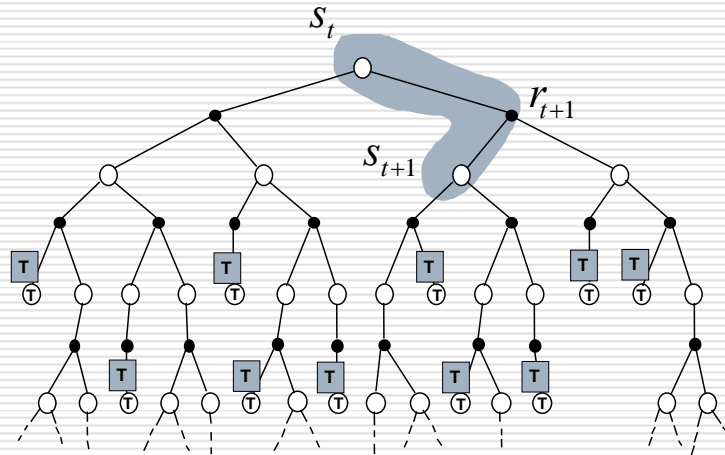
$$V^{\pi}(s_t) = E\{R_t | s_t, \pi\} = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+1+k} | s_t, \pi\right\}$$



12

Temporal difference

$$\hat{V}_{t+1}^\pi(s_t) = \hat{V}_t^\pi(s_t) + \alpha(r_{t+1} + \gamma \hat{V}_t^\pi(s_{t+1}) - \hat{V}_t^\pi(s_t))$$



15

Q-Learning (1)

- Watkins, 1989
- Can be seen as a reinforcement learning algorithm
- It is one of the most used
- Starting from the Q^* function of the Bellman equations

$$Q^*(s, a) = \sum_{s' \in S} p_{ss'}^a \{R_{ss'}^a + \gamma V^*(s')\}$$

$$V^*(s) = \max_{a \in A} [Q^*(s, a)]$$

16

Q-Learning (2)

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

```
Initialize  $Q(s, a)$  arbitrarily
Repeat (for each episode):
  Initialize  $s$ 
  Repeat (for each step of episode):
    Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
    Take action  $a$ , observe  $r, s'$ 
     $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'$ 
  until  $s$  is terminal
```

17

Q-Learning (3)

→ From theory to practice

□ Algorithm

```
while(true)
  input = getInput()
  action = chooseAction(input, qvalues)
  do(action)
  qvalues = update(qvalues, getInput())
```

- `getInput()`: get perception from environment
- `chooseAction(...)`: choose the action finding the maximum reward
- `do(...)` execute the action chosen
- `update(...)`: update qvalues

18

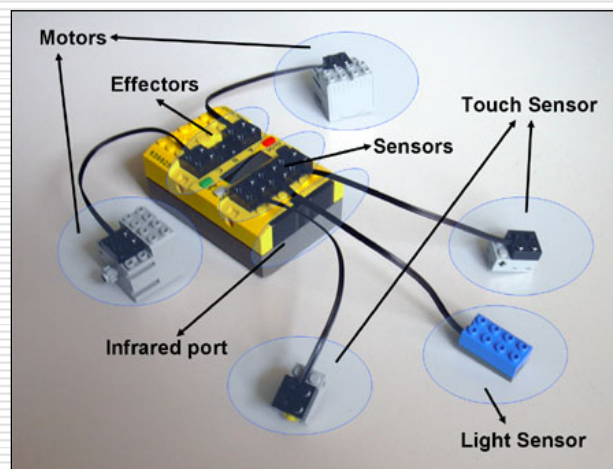
Q-Learning (4)

- input = getInput()
 - Find the new state of the robot
- The states'table can be used by the function $Q(\text{state}, \text{action})$
- Es. $Q(2, 3) = 3.2$
- For each step the robot will be in a new state "x" where it will execute the action with the maximum value $Q(x, \text{azione})$

	Act 1	Act 2	Act 3
State1	1.3	4.2	-0.3
State2	0.2	1.2	3.2

19

Lego Mindstorms?



20

Which development tool?

Name	Language	Must change firmware	Notes
RIS	"Visuale"	NO	Default Software, simple to use with simple problem
LEGO Mindstorm SDK	LASM, Mindscript, ActiveX complaint	NO	SDK distributed by LEGO
NQC	Syntax similar to C	NO	The best thing if you do not want to change firmware
LegOS	C	YES	Not so simple to use, focus on performances
LeJOS	Java	YES	Good performances, simple to use

21

Robotics application (1)

- A robot that find the light inside a room
- Action
 - Two motors
 - Each motor have three actions:
 - FORWARD, BACKWARD, STOP
 - Since we are using two motors $3*3 = 9$ actions
 - Motor1.stop() && Motor2.stop()
 - Motor1.stop() && Motor2.forward()
 - ...
 - Motor1.backward() && Motor2.backward()

Act1	Act2	Act3	Act4	Act5	Act6	Act7	Act8	Act9
------	------	------	------	------	------	------	------	------

22

Robotics application (2)

- A robot that find the light inside a room
- States
 - Light sensor
 - We simulta the use of two sensors in this way:
 - Left Light > Right Light
 - Left Light < Right Light
 - Left Light == Right Light
 - Front touch sensor
 - Pressed / Unpressed
 - Rear touch sensor
 - Pressed / Unpressed
 - Last action repeted 5 times
 - Yes / No
 - Counting $3*2*2*2 = 24$ states

23

Robotics application(3)

- Summarizing
 - 9 Actions, 24 States
 - Total of 216 Q-Values

	Act1	...	Act9
State1	Q(1,1)	Q(x,1)	Q(9,1)
...	Q(1,y)	Q(x,y)	Q(9,y)
State24	Q(1,24)	Q(x,24)	Q(9,24)

- **How to update the Q-Values?**
 - Reward due to light =
actual light – previous light
 - Reward due to collision = -2
 - Reward due to repeted action (> 5) = -3

24

Robotics application(4)

- How to update the Q-Values in detail?
- If we assign the Q-Value from the reward it will be too drastic and this will not assure good error-tolerant behaviour
- We should use something that will change slowly the Q-Values
 - $Q_{\text{new}}(s,a) = Q_{\text{old}}(s,a) + \alpha(R(s) + k)$
 - α is the learning rate ($0 \leq \alpha < 1$)
 - k can be setted to 0 for semplicity (now)

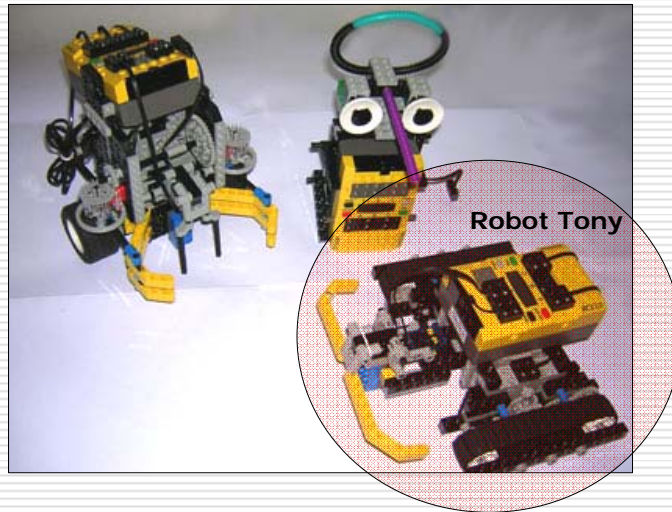
25

Robotics application(5)

- $Q_{\text{new}}(s,a) = Q_{\text{old}}(s,a) + \alpha(R(s) + k)$
- If $k=0$, it will update the value only considering the reward and not considering the new Q-Value
- If $k = \gamma(\max(Q(j, a1) - Q_{\text{old}}(s,a))$
- Where $(j, a1)$ are the new action and the new state

26

Implementation (1)



27

Implementation (2)

- Using LeJOS
- Three classes
 - TonyTank
 - Main class, cycle of perception, thinking and action.
 - TonyBrain
 - Class where we have implemented the learning function with the update of Q-Values
 - TonyMisc
 - Class with misc methods
 - We also used a value called RANDOM_EXPLORATION that force the robot to do a random action after n-eras.
 - This is due to fact that the robot could be stuck in some local minimum situation

28

Implementation TonyTank.java

□ Main file

```
byte actions = 9;
byte states = 24;
TonyMisc tonyMisc = new TonyMisc(states, actions);
TonyBrain q = new TonyBrain(states, actions, util);
byte[] e = new byte[5]; // e represents the environment
byte [] motorCommand = new byte[2]; // Commands to send to motors

Sensor.S2.setTypeAndMode (SENSOR_TYPE_LIGHT, SENSOR_MODE_PCT);
Sensor.S2.activate();

while(true) {
    getValueFromEnvironment(e);

    byte a = q.getAction(e);

    tonyMisc.getCommands(a, motorCommand);

    performAction(motorCommand);
}
```

29

Implementation TonyBrain.java

□ General learning constants

```
public static float LEARN_RATE = 0.45f;
public static float RANDOM_EXPLORATION =
    0.15f;
```

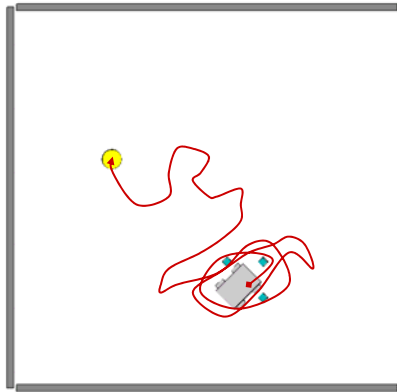
□ float [][] Q; // table of Q-values (state,action)

□ The core (getAction)

```
r = util.getRewardValue(e);
a1 = getMaxAction(j);
Q[i][a] = Q[i][a] + LEARN_RATE * (r + Q[j][a1] -
    Q[i][a]); //setted  $\gamma = 1$  for semplicity
```

30

Dynamic behaviour



31

Conclusions (1)

- Reinforcement learning used often in robotics
 - Environment exploration dell'ambiente
 - Actuators
- Coupled interaction with the environment
 - The applicability in this context relies upon the precision of sensors
- Discrete states requirement
 - Propose in future the use of fuzzy control insted

32

Conclusions (2)

- It partially resolves the problems of the supervised learning without a training set
- It partially resolve the problems of the classic unsupervised learning
 - ...which can classify datas but did not provide an action to execute
- Examples of robotics applications
 - S. Schaal and Christopher Atkeson. Robot juggling: An implementation of memory-based learning. *Control Systems Magazine*, 14, 1994. → Control theory
 - Sridhar Mahadevan and Jonathan Connell. Automatic programming of behavior-based robots using reinforcement learning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, Anaheim, CA, 1991. → Mobile robotics
 - Maja J. Mataric. Reward functions for accelerated learning. In W. W. Cohen and H. Hirsh, editors, *Proceedings of the Eleventh International Conference on Machine Learning*. Morgan Kaufmann, 1994. → Since the problem is complex, they use the q-learning on a very big dimensional space

33

References (1)

- Sito del corso di Robotica 2
Prof. Gini
<http://www.elet.polimi.it/upload/gini/robot2/index.html>
- LEGO Mindstorms
<http://www.legomindstorms.com>
- LEGO Mindstorms detailed slides,
from the same author
<http://www.marcocioffi.com/archives/category/interests/robotics/>

34

References (2)

- R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.
- J. Wyatt, Reinforcement Learning: A Brief Overview. *Perspectives on Adaptivity and Learning*. Springer Verlag, 2003.
- L.Kaelbling, M.Littman and A.Moore, Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237-285, 1996.