Learning For Robotics:

Introduction & Outline

Nguyen Sao Mai
U2IS, ENSTA
http://nguyensmai.free.fr

COURSE OUTLINE

1. INTRODUCTION
   - 1.1. What is Robot Learning ?
   - 1.2. Interaction with tutors : imitation learning
   - 1.3. Interaction with the environment : Reinforcement Learning

REFERENCES

1. BOOKS/BLOGS
   - 1.1. Aude Billard, Sylvain Calinon, Rüdiger Dillmann, Stefan Schaal, Ch 59 Robot Programming by Demonstration in : Siciliano, Bruno, and Oussama Khatib, eds. Springer handbook of robotics. Springer, 2016.
   - 1.2. R. S. Sutton and A. G. Barto. Reinforcement Learning: an introduction. MIT Press, 2018. http://www.incompleteideas.net/
   - 1.3. https://mpatacchiola.github.io/blog/
2. OTHER COURSES
   - 1.1. Advanced Robotics Peter Abbeel https://people.eecs.berkeley.edu/~pabbeel/cs287-fa19/
   - 1.2. Reinforcement Learning by Poupart : https://cs.uwaterloo.ca/~ppoupart/teaching/cs885-spring18/schedule.html
   - 1.3. https://www.coursera.org/learn/robotics-learning#syllabus
   - 1.4. deep RL http://rail.eecs.berkeley.edu/deeprlcourse/ and https://www.youtube.com/watch?v=e2PpdPC34kI&list=PL_iWQOsE6TfURIIhCrlt-wj9ByIVpbfGc&index=9
3. SIMULATION ENVIRONMENT :
   - 3.1. gym.openai.com/

EVALUATION

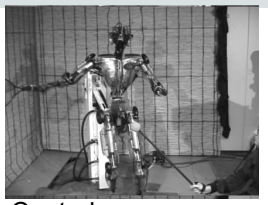
1. PRESENTATION + Q&A
2. QUIZZ
3. PROJECT

Sao Mai Nguyen
U2IS, ENSTA
http://nguyensmai.free.fr

1. What is Robot Learning ?

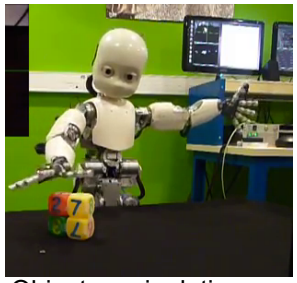
1. WHAT IS ROBOT LEARNING ?
1.1. Robotics Fields

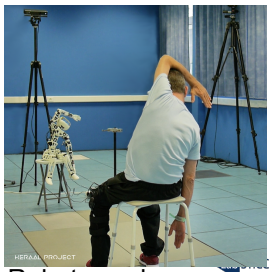

Navigation


Control


Social robotics

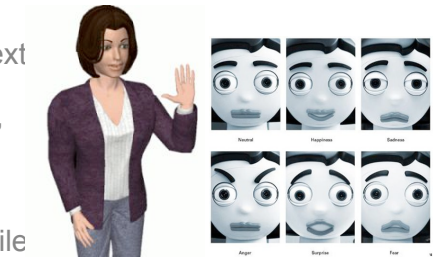

Teleoperation

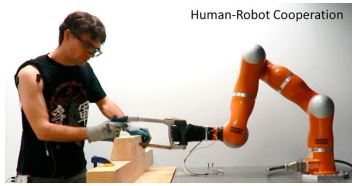

Object manipulation


Robot coach

1. WHAT IS ROBOT LEARNING ?
1.2. Interaction with the environment

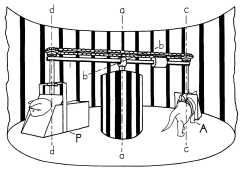
- **Vocal interaction**: speech recognition, speech generation (text to-speech)
- **Natural interaction** : multi-modal, non-verbal interaction, gesture, expressive emotion-based interaction
- **Physical interaction** : touch (tactile sensors), grasping, manipulation
- **Socio-cognitive skills** : socially acceptable behaviours, turn-taking, coordination, theory of mind




Human-Robot Cooperation

1. WHAT IS ROBOT LEARNING ?
1.3. Theoretical Approaches to Robot Learning

- **Developmental approaches** : there is an orderly way to learn multiple tasks, the learning is progressive and hierarchical -> Developmental psychology
- **Cognitive approaches** : inspired by cognitive science, neuroscience, neuronal computation models. Decomposes into a task into cognitive skills/functions
- **Life-long learning : the environment and tasks can change**
- **Embodiment** : the environment has a physical incarnation, the agent has a physical incarnation => its learning, capacities, behaviour depends on its physical body **https://www.youtube.com/watch?v=3FlzxKuqzUM**
- **Enactivism** : Learning of the agent in its environment


(Held and Hein, 1963)

1. WHAT IS ROBOT LEARNING ?

1.4. Computational Approaches to Robot Learning

- **Active learning** : select data that improves the learning the most; select actions that maximize learning performance
- **Interactive perception** : observing the outcomes of different manipulation actions
- **Transfer learning** : extracts knowledge from one or more source tasks and applies the knowledge to a target task. Knowledge from a source task is used in a target task to speed up learning.
- **Multi-task learning** : learn several tasks simultaneously

1. WHAT IS ROBOT LEARNING ?

1.5 Human-Robot Interaction (HRI) : Definition

Goal: understanding, designing, and evaluating robotic systems for use by or with humans.
Interaction : requires communication between robots and humans.

2 categories of communication:

- Remote interaction – The human and the robot are not co-located and are separated spatially or even temporally.
- Proximate interactions – The humans and the robots are co-located.

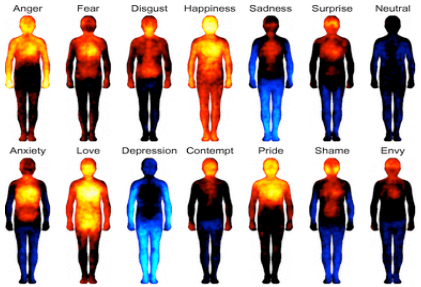
HRI, the multi-disciplinary field started to emerge in the mid 1990s and early years of 2000.
A multi-disciplinary approach:
- Robotics
- cognitive science
- human factors
- natural language
- psychology,
- human-computer interaction

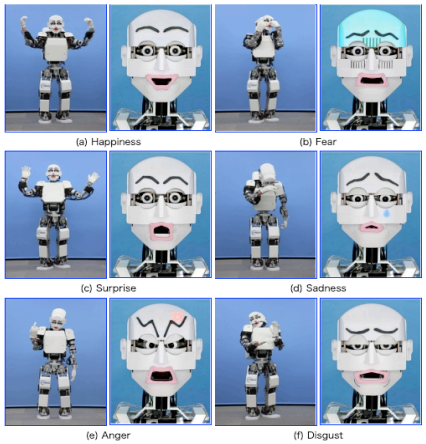
3 very influential areas :
- robot-assisted search and rescue
- space exploration
- assistive robots

1. WHAT IS ROBOT LEARNING ?

1.6 Human-Robot Interaction (HRI) : Emotion classification

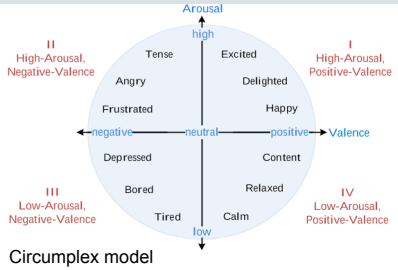
Primary emotions:
happiness, sadness, disgust, fear, surprise, and anger
pride, shame, embarrassment, and excitement



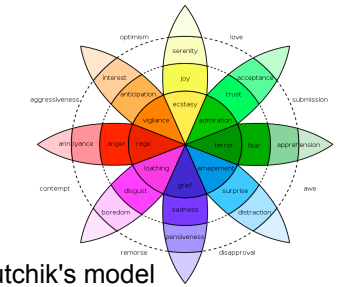


1. WHAT IS ROBOT LEARNING ?

1.6 Human-Robot Interaction (HRI) : Emotion classification

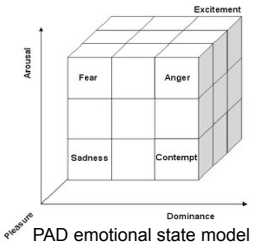


Circumplex model

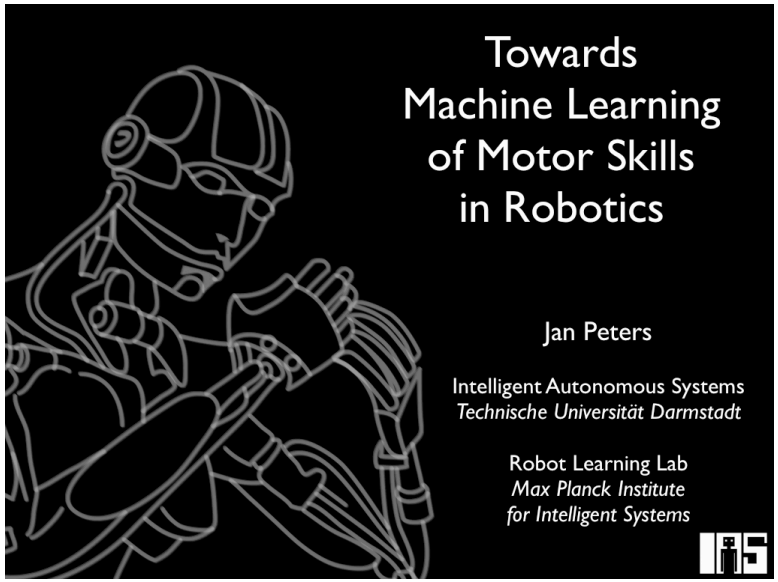


Plutchik's model

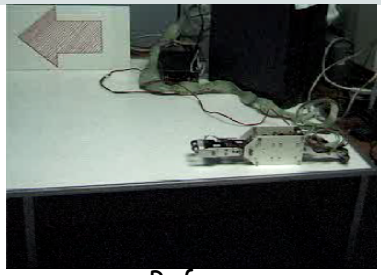
Dimensional emotions
- circumplex model
- the vector model
- the Positive Activation – Negative Activation (PANA) model
- Plutchik's model
- PAD emotional state model

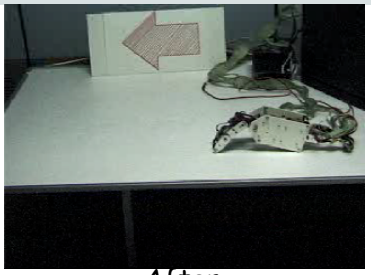


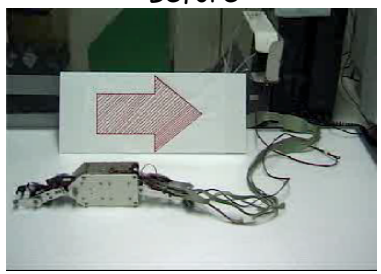
PAD emotional state model

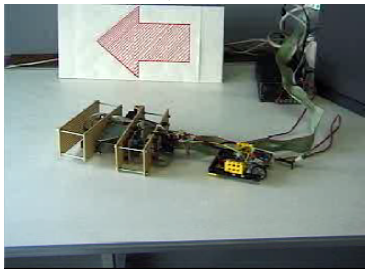


3:13


Before

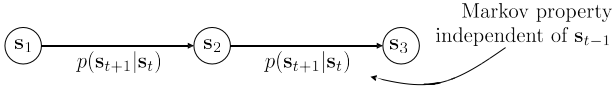

After


Backward


New Robot, Same algorithm

Boston dynamic spot : https://youtu.be/6Zbhvaac68Y

MARKOV PROCESSES



Andrey Markov

A Markov Process is defined by a set of $\{\mathcal{S}, \mathcal{T}\}$ where :

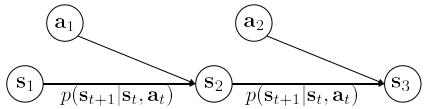
$\mathcal{S}$ is the state space (discrete or continuous), and states $s \in \mathcal{S}$

$\mathcal{T}$ is the transition operator, $\mathcal{T} : S \times S \rightarrow [0, 1]$ $T_t(s, s') = P(s_{t+1} = s' | s_t = s)$

why "operator"? let $\mu_{t,i} = p(s_t = i)$ $\vec{\mu}_t$ is a vector of probabilities

let $\mathcal{T}_{i,j} = p(s_{t+1} = i | s_t = j)$ then $\vec{\mu}_{t+1} = \mathcal{T}\vec{\mu}_t$

MARKOV DECISION PROCESS

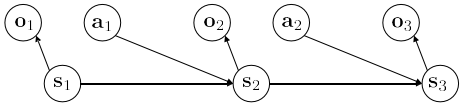


Richard Bellman

A Markov Decision Process (MDP) is defined by a set of $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, R, H, \gamma\}$ where :

$\mathcal{S}$ is the state space (discrete or continuous),	and states $s \in \mathcal{S}$	
$\mathcal{A}$ is the action space (discrete or continuous),	and action $a \in \mathcal{A}$	
H the horizon over which the agent will act,	$H \in \mathbb{N}$	
$\mathcal{T}$ is the transition operator,	$\mathcal{T} : S \times A \times S \times \{0, 1, ..., H\} \rightarrow [0, 1]$	$T_t(s, as') = P(s_{t+1} = s' \| s_t = s, a_t = a)$
R is the reward function	$R : S \times A \times S \times \{0, 1, ..., H\} \rightarrow \mathbb{R}$	$R_t(s, as')$ gives the immediate reward received after transitioning from state s to state s' due to action a
γ the discount factor,	$\gamma \in [0, 1]$	

POMDP

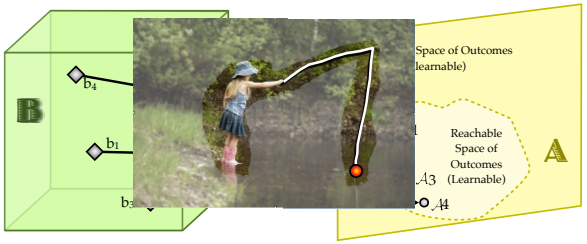


A Partially Observable Markov Decision Process (POMDP) is defined by a set of $\{\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \epsilon, R, H, \gamma\}$ where :

$\mathcal{S}$ is the state space (discrete or continuous),	and states $s \in \mathcal{S}$	
$\mathcal{A}$ is the action space (discrete or continuous),	and action $a \in \mathcal{A}$	
$\mathcal{O}$ is the observable space (discrete or continuous),	and observables $o \in \mathcal{A}$	
H the horizon over which the agent will act,	$H \in \mathbb{N}$	
$\mathcal{T}$ is the transition operator,	$\mathcal{T} : S \times A \times S \times \{0, 1, ..., H\} \rightarrow [0, 1]$	$T_t(s, as') = P(s_{t+1} = s' \| s_t = s, a_t = a)$
$\mathcal{E}$ is the emission function,		$\mathcal{E}(o_t, s_t) = p(o_t \| s_t)$
R is the reward function	$R : S \times A \times S \times \{0, 1, ..., H\} \rightarrow \mathbb{R}$	$R_t(s, as')$ gives the immediate reward received after transitioning from state s to state s' due to action a
γ the discount factor,	$\gamma \in [0, 1]$	

1. WHAT IS ROBOT LEARNING ?

Formalisation of the Robot Learning Problem



Robot control: Learn a probabilistic distribution p(b | a)
Given a goal position, which arm movement should I perform?

- The agent does not have a model of the environment
- A and B: continuous spaces, of high dimension : **large search space**
- Stochasticity, redundancy
- Inhomogeneous (unlearnability)

2. INTERACTIONS WITH TUTORS:

IMITATION LEARNING

2. IMITATION LEARNING
2.1. What to imitate ?

Imitation learning
Learning from demonstation
Programming by demonstration
Behavioral cloning

Mimicry : reproduce the movement

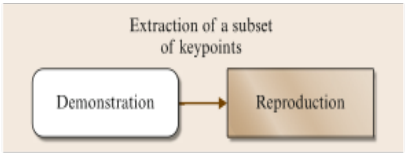
Emulation : reproduce the effects/outcomes

2. IMITATION LEARNING
2.2. Why imitation learning? What is imitation learning?

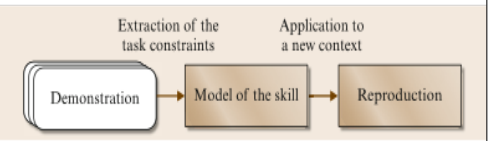
- An implicit, *natural* means of training a machine that would be **accessible to lay people**
- A powerful mechanism for **reducing the complexity of search** spaces for learning
- Studying and modeling the **coupling of perception and action**

2. IMITATION LEARNING
2.2. Why imitation learning? What is imitation learning?

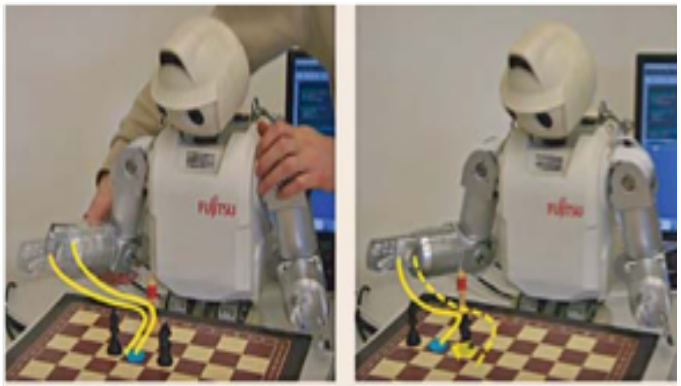
Copying the demonstrated movements



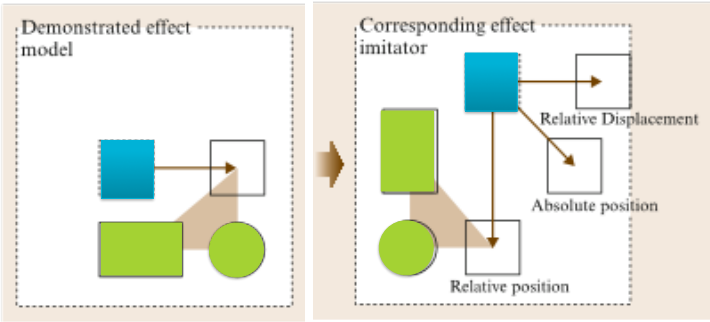
Generalize across sets of demonstrations.

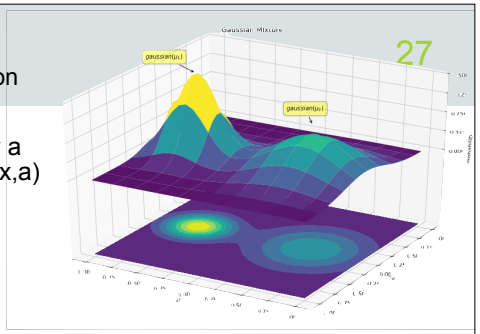


- How to **generalize** a task
- How to **evaluate** a reproduction attempt
- How to better define the role of the **user** during learning



- **Metric of imitation performance:** extract the important features characterizing the skill
- An **optimal controller** to imitate by trying to **minimize this metric**





- We can model observed data X= (x,a) by a probabilistic density distribution P(X) = p(x,a)
  - Gaussian Mixture Models:

$$p(\boldsymbol{X}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{K} \pi_i \mathcal{N}(\boldsymbol{X}, \boldsymbol{\mu_i}, \boldsymbol{\Sigma_i})$$

  - Multivariate Gaussian

$$\mathcal{N}(\boldsymbol{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right)$$

  $\boldsymbol{\mu}$ is the mean

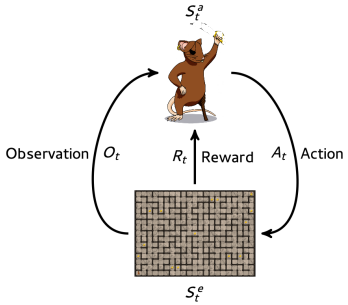
  $\boldsymbol{\Sigma}$ is the covariance matrix

- We can infer the robotic command
  - v= argmax$_v$ p(v|x)

3. INTERACTION WITH THE ENVIRONMENT :

REINFORCEMENT LEARNING

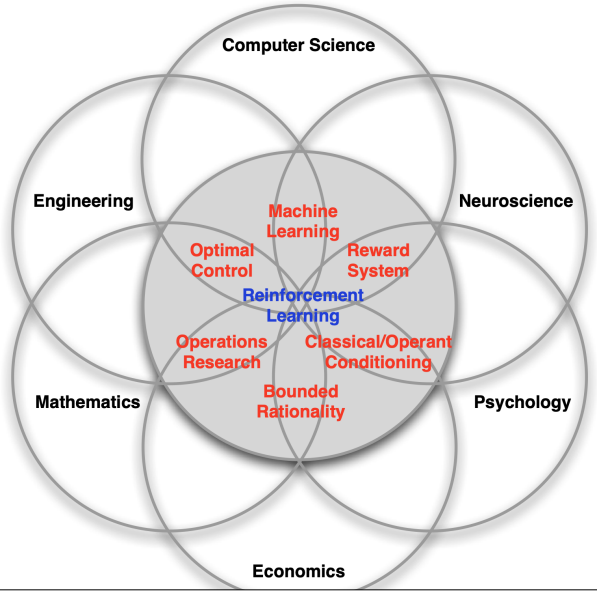
3. REINFORCEMENT LEARNING
3.3. Definitions



- The agent …
  - performs action A_t
  - obtains an observation O_t
  - obtains reward R_t
- The environment …
  - receives action At
  - produces Ot
  - produces reward Rt

- Agent seeks to maximize its cumulative **reward** on the long run
- Agent learns a policy **mapping states to actions**

REINFORCEMENT LEARNING
What is reinforcement learning?

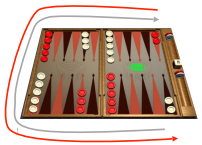


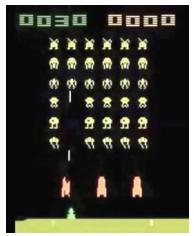
✤Psychology : classical conditioning, Pavlovian control. Pavlov found that an animal's innate responses to certain triggering stimuli can come to be triggered by other stimuli that are quite unrelated to the inborn triggers (1927)
✤Neuroscience : reward prediction error hypothesis of dopamine neuron activity : Dopamine appears to convey temporal-difference (TD) errors (1996).

David Silver 2015

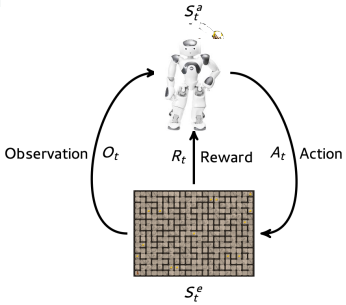
REINFORCEMENT LEARNING
Some Reinforcement learning SUCCESSES

✤Learned the world's best player of Backgammon (Tesauro 1995)
✤Learned acrobatic helicopter autopilots (Ng, Abbeel, Coates et al 2006+)
✤Widely used in the placement and selection of advertisements and pages on the web (e.g., A-B tests)
✤Used to make strategic decisions in *Jeopardy!* (IBM's Watson 2011)
✤Achieved human-level performance on Atari games from pixel-level visual input, in conjunction with deep learning (Google Deepmind 2015)
✤Google Deepmind's AlphaGo defeats the world Go champion, vastly improving over all previous programs (2016)
✤In all these cases, performance was better than could be obtained by any other method, and was obtained without human instruction





Definitions



Markov Decision Process
- Set of states S
- Set of actions A
- Transition model $Pr(s_t|s_{t-1}, a_{t-1})$
- Reward model $R(s_t,a_t)$
- Discount factor γ
- Horizon (nb of time steps) : h

- Agent seeks to maximize its cumulative **reward** on the long run
- Environment may be unknown, nonlinear, stochastic and complex and non-observable :
  - Full observability : $S_t^a = S_t^o = O_t$
  - Partial observability: s_t^a is estimated by the environment

Goal of RL

The objective is to maximize long-term future reward
That is, to choose A_t so as to maximize $R_{t+1}, R_{t+2}, R_{t+3}, \ldots$
But what exactly should be maximized?
The discounted *return* at time t:

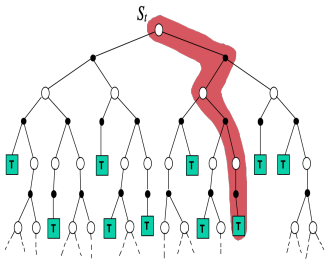
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \qquad \gamma \in [0, 1)$$

γ	Reward sequence	Return
0.5(or any)	1 0 0 0…	
0.5	0 0 2 0 0 0…	
0.9	0 0 2 0 0 0…	
0.5	-1 2 6 3 2 0 0 0…	

REINFORCEMENT LEARNING
Policy and Value Function

Policy π

- A policy is the agent behavior
- Map from state to action
- Deterministic : $a = \pi(s)$
- Stochastic : $\pi(a|s) = P[A_t = a|S_t = s]$



backup diagram

Value Function V

- Prediction of future reward
- Evaluates the goodness of states
- Action selection using the value function
- $V_\pi(s) = \mathbb{E}_\pi[\sum_k \gamma^k r^{t+k+1} | s_t = s]$

Q-Value Function Q

- same as V but for each action : prediction of future reward
- Evaluates the goodness of state-action pairs
- Action selection using the value function
- $Q_\pi(s, a) = \mathbb{E}_\pi[\sum_k \gamma^k r^{t+k+1} | s_t = s, a_t = a]$

REINFORCEMENT LEARNING
3.5. The Bellman equation

- Optimal solution
  - Policy π*
  - V*(s) = $\max_\pi V_\pi(s)$
  - Q*(s,a)= $\max_\pi Q_\pi(s,a)$

- Bellmann equation: from state s_t=s,a,r_{t+1} and next state s_{t+1}

$$V_\pi(s) = \mathbb{E}_\pi\{r_{t+1} + \gamma V_\pi(s_{t+1}) | s_t = s\} = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$$

$$Q_\pi(s, a) = \mathbb{E}_\pi\{r_{t+1} + \gamma V_\pi(s_{t+1}) | s_t = s, a_t = a\} = \sum s' p(s, r|s, a)(r + \gamma \sum_{a'} \pi(a'|s') Q_\pi(s', a'))$$

$$V^*(s) = max_a \mathbb{E}\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\} = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V^*(s')]$$

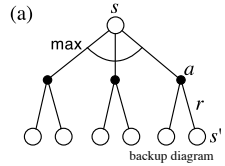
$$Q^*(s, a) = \mathbb{E}_\pi\{r_{t+1} + \gamma max_{a'} q^*(s_{t+1}, a') | s_t = s, a_t = a\} = \sum_{s',r} p(s', r|s, a)[r + \gamma max_{a'} Q^*(s', a')]$$

- Bellman optimality equation expresses the fact that the value of a state under an optimal policy must equal the expected return for the best action from that state

Bellman Optimality Equation for v_* and q_*

The value of a state under an optimal policy must equal the expected return for the best action from that state:
v_* is the unique solution of this system of non-linear equations.

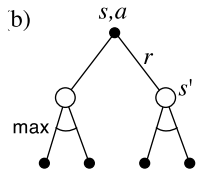
$$\begin{aligned} v_*(s) &= \max_a q_{\pi_*}(s, a) \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma v_*(s')\big]. \end{aligned}$$



backup diagram

q_* is the unique solution of this system of nonlinear equations.

$$\begin{aligned} q_*(s, a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a\Big] \\ &= \sum_{s',r} p(s', r|s, a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big]. \end{aligned}$$



backup diagram

Many RL methods can be understood as approximately solving the Bellman Optimality Equation.

MAIN CHARACTERISTICS

Copier / coller les textes

- Reinforcement learning methods specify how the agent changes its policy as a result of experience.
- Roughly, the agent's goal is to get as much reward as it can over the long run.

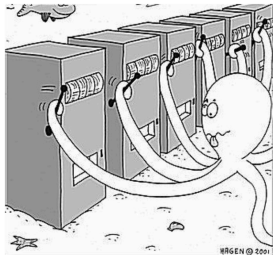
Signature challenges of RL

- Evaluative feedback (reward)
- Sequentiality, delayed consequences
- Need for trial and error, to explore as well as exploit:
  - Exploration and Exploitation Dilemna : Repeat with existing strategy (Exploitation) or try a new strategy (Exploration) ?
- Non-stationarity
- The fleeting nature of time and online data

EXPLORATION EXPLOITATION DILEMMA

We can learn by using non-optimal policies

- Online decision-making involves a fundamental choice:
  - **Exploitation** Make the best decision given current information
  - **Exploration** Gather more information; discover new potentially better solutions
- The best long-term strategy may involve short-term sacrifices
- Gather enough information to make the best overall decisions



Examples ?

CREDIT ASSIGNMENT PROBLEM

The reward only indicates how valuable a sequence of states, actions is.
The reward can come late => Temporal credit assignment
If the agent has many parts => Spatial credit assignment

- Which actions along the path were responsible for getting the reward and to what extent.

Examples ?

VALUE BASED METHODS

Bellman equation

Monte Carlo sampling

$$\begin{aligned} v_*(s) &= \max_a q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma v_*(s')\big]. \end{aligned}$$

TD methods combine Monte Carlo + Dynamic Programming :

- sampling: replace expectation by samples

$$V(S_t) \leftarrow V(S_t) + \alpha\big[G_t - V(S_t)\big]$$

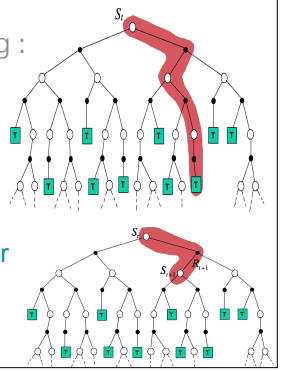
target: the actual return after time t

- bootstrapping: replace a real value by an estimate

$$V(S_t) \leftarrow V(S_t) + \alpha\big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big]$$

TD error

target: an estimate of the return

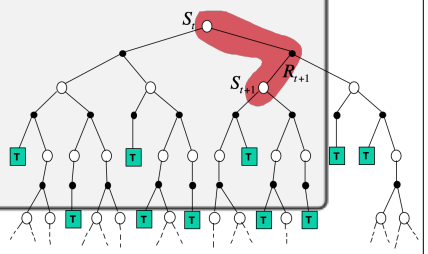


Tabular TD(0) for estimating v_π

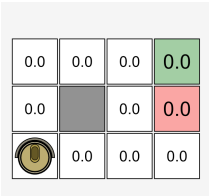
Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

Time Difference TD error



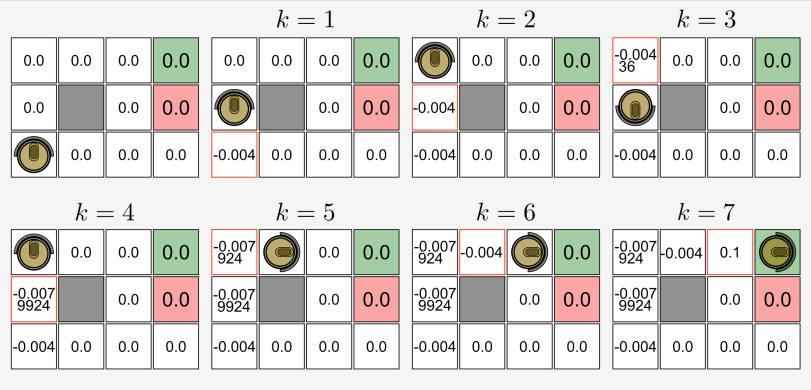
TD PREDICTION EXAMPLE



The robot is in a 4x3 world with an unknown transition model. The only information about the environment is the states availability. Since the robot does not have the reward function it does not know which state contains the charging station (+1) and which state contains the stairs (-1).
γ=0.9, α=0.1

X

3.6.



Bellman equation

$$q_*(s,a) = \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a\Big]$$

Sampling

$$= \sum_{s',r} p(s', r|s, a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big].$$

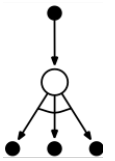
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

(Watkins, 1989)



Technical requirements.

- All states and actions are visited infinitely often
- Learning rate schedule such that for all state and action pairs (s,a):

$$\sum_{t=0}^{\infty} \alpha_t(s,a) = \infty \qquad \sum_{t=0}^{\infty} \alpha_t^2(s,a) < \infty$$

How to sample actions? off-policy

- **ε**-Greedy: choose random action with prob.**ε**, otherwise choose action greedily

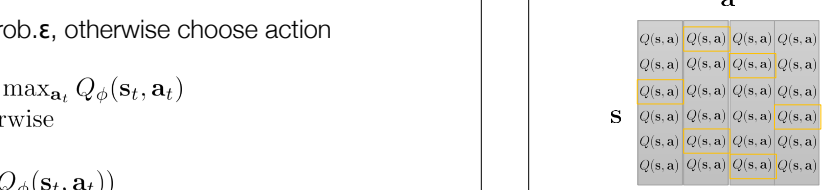
$$\pi(\mathbf{a}_t|\mathbf{s}_t) = \begin{cases} 1 - \epsilon \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q_\phi(\mathbf{s}_t, \mathbf{a}_t) \\ \epsilon/(|\mathcal{A}| - 1) \text{ otherwise} \end{cases}$$

- Botzmann exploration

$$\pi(\mathbf{a}_t|\mathbf{s}_t) \propto \exp(Q_\phi(\mathbf{s}_t, \mathbf{a}_t))$$

Tabular representation

- discrete action, state spaces
- Curse of dimensionality: the computational requirements grow exponentially with the number of state variables



$\arg\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a}) \rightarrow$ policy

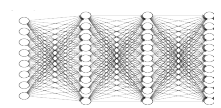
https://youtu.be/kEnAvPg61Ps

Approximate Q-Learning

- discrete or continuous action, state spaces
- Parametrised function Q_ϕ(s,a)
  - Linear function in features

$$Q_\phi(s,a) = \phi_0 f_0(s,a) + \phi_1 f_1(s,a) + ... + \phi_n f_n(s,a)$$

  - Neural net



$Q_\phi(\mathbf{s}, \mathbf{a})$

  - decision tree….