Reinforcement Learning

Michèle Sebag ; TP : Diviyan Kalainathan
TAO, CNRS − INRIA − Université Paris-Sud

Nov. 13th, 2017

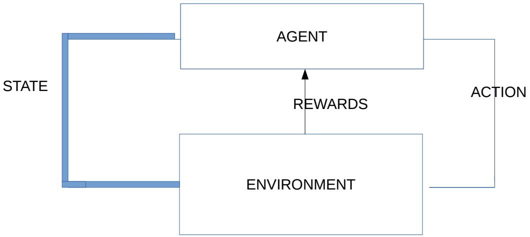
Credit for slides: R. Sutton, F. Stulp

Types of Machine Learning problems

WORLD – DATA – USER

Observations	+ **Target**	+ Rewards
Understand **Code**	**Predict** **Classification/Regression**	Decide Action Policy/Strategy
Unsupervised **LEARNING**	**Supervised** **LEARNING**	Reinforcement LEARNING

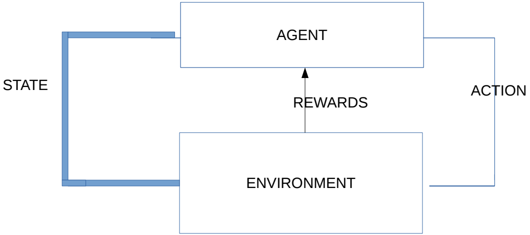
Reinforcement Learning



Position of the problem

- An agent, spatially and temporally situated
- Stochastic and uncertain environment
- Goal: select an action in each time step,
- ... in order maximize expected cumulative reward over a time horizon

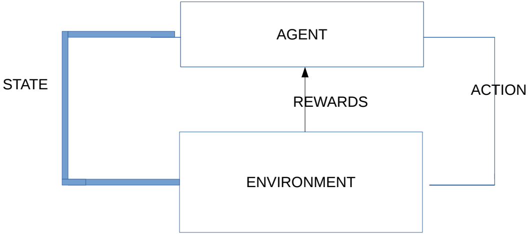
Reinforcement Learning



Features

- dimension(situation) $\gg$ dimension(action) $>$ dimension(reward) $= 1$
- Rewards:
  - enable the agent to evaluate itself in an autonomous way
  - with **delay**
  - only evaluate what the agent did (not what could/should be done)
- Exploration / Exploitation dilemma

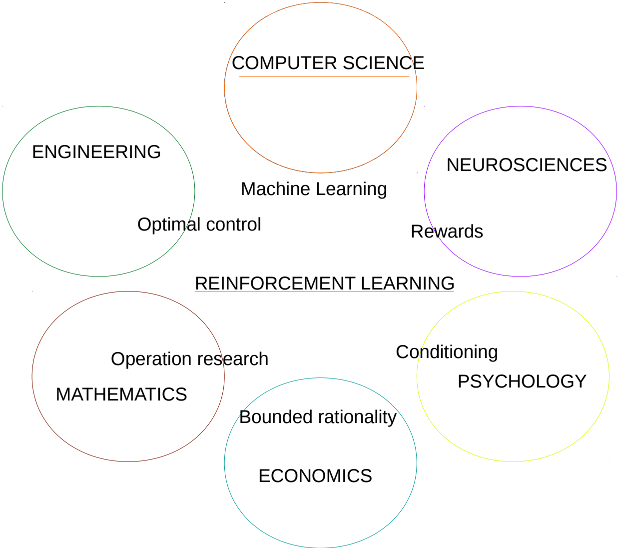
Reinforcement Learning



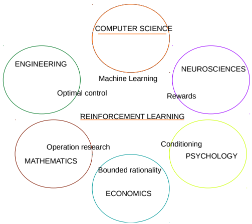
Features

- dimension(situation) $\gg$ dimension(action) $>$ dimension(reward) $= 1$
- Rewards:
  - enable the agent to evaluate itself in an autonomous way **create its data**
  - with **delay**
  - only evaluate what the agent did (not what could/should be done)
- Exploration / Exploitation dilemma

Reinforcement Learning & neighbor disciplines

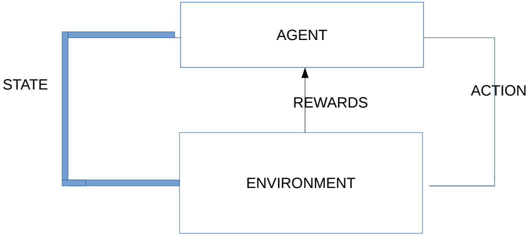


Reinforcement Learning, terms



RL	synonyms
state	situation, stimulus
action	control, response
reward	payoff, gain, cost

Reinforcement Learning, output



What is learned ?

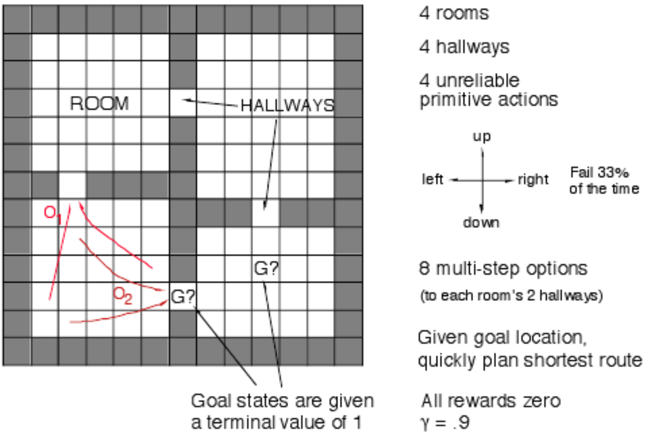
A policy = strategy = { state $\mapsto$ action }

Reinforcement Learning

Context

An unknown world.
Some actions, in some states, bear rewards with some delay [with some probability]

Goal : **find policy (state $\rightarrow$ action) maximizing the expected reward**



The Reinforcement Learning AIC Master Module

Contents

1. Position of the problem, vocabulary

   Markov Decision Process, policy, expected returns

2. Finite case

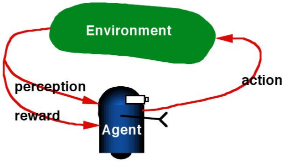
   Dynamic Programming; Value estimation, Monte Carlo, TD

3. Infinite case

   Function approximation, direct policy search

4. More

   Inverse RL, Multi-armed bandits, Deep RL



Evaluation

- Project
- Exam
- Voluntary contributions (15mn talks, adding resources)

Pointers

- "The Book"
  - "Reinforcement Learning: An Introduction", R. Sutton and A. Barto
  - On-line at: `http://webdocs.cs.ualberta.ca/~sutton/book/the-book.html`
- Slides on-line at
  - https://tao.lri.fr/tiki-index.php?page=Courses
  - http://perso.ensta-paristech.fr/∼stulp/aic/
  - https://www.microsoft.com/en-us/research/video/tutorial-introduction-to-reinforcement-learning-with-function-approximation/

Suggested:

- See videos *before* the course
- Let's discuss during the course what is unclear, what could be done otherwise, what could be done.

Overview

Introduction

Position of the problems

Formalisation: the value function

Values

Facets

Disciplines

- Machine Learning
- Economics
- Psychology
- Neurosciences
- Control
- Mathematics

Means

- Rewards, incentives, conditioning
- Bounded rationality

Reinforcement Learning

Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will – others things being equal – be more firmly connected with the situation, so that when it recurs, they will more likely to recur;

those which are accompanied or closely followed by discomfort to the animal will – others things being equal – have their connection with the situation weakened, so that when it recurs, they will less likely to recur;

the greater the satisfaction or discomfort, the greater the strengthening or weakening of the link.

Thorndike, 1911

Formal background

Notations

- State space $\mathcal{S}$
- Action space $\mathcal{A}$
- Transition model $p(s, a, s') \mapsto [0, 1]$
- Reward $r(s)$

Goal

- Find policy $\pi : \mathcal{S} \mapsto \mathcal{A}$

$$\text{Maximize } E[\pi] = \text{ Expected cumulative reward}$$

(detail later)

Applications

- **Robotics**
  Navigation, football, walk, ...
- **Control**
  Helicopter, elevators, telecom, smart grids, manufacturing, ...
- **Operation research**
  Transport, scheduling, ...
- **Games**
  Backgammon, Othello, Tetris, Go, ...
- **Other**
  Computer Human Interfaces, ML (Feature Selection, Active learning, Natural Language Processing,...)

Myths

1. Pandora (the box)
2. Golem (Praga)
3. The chess player (The Turc)
   Edgar Allan Poe
4. Robota
5. Movies...

Myths

1. Pandora (the box)
2. Golem (Praga)
3. The chess player (The Turc)
   Edgar Allan Poe
4. Robota
5. Movies... Metropolis, 2001 Space Odyssey, AI, Her, ...

Types of robots: 1. Manufacturing

∗closed world, target behavior known
∗task is decomposed in subtasks
∗subtask: sequence of actions
∗no surprise

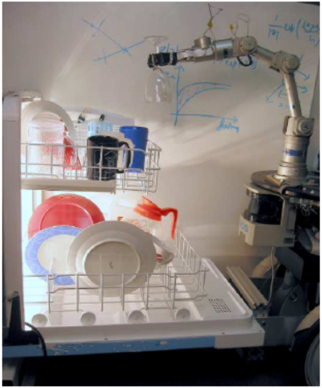
Types of robots: 2. Autonomous vehicles

∗open world
∗task is to navigate
∗action subject to precondition

Types of robots: 2. Autonomous vehicles

∗a wheel chair
∗controlled by voice
∗validation ?

Types of robots: 3. Home robots



open world

sequence of tasks

each task requires navigation and planning

Robot

https://www.microsoft.com/en-us/research/video/tutorial-introduction-to-reinforcement-learning-with-function-approximation/
5:14

Vocabulary 1/3

- **State of the robot** set of states $\mathcal{S}$
  A state: all information related to the robot (sensor information; memory)
  Discrete ? continuous ? dimension ?

- **Action of the robot** set of actions $\mathcal{A}$
  values of the robot motors/actuators.
  e.g. a robotic arm with 39 degrees of freedom.
  (possible restrictions: not every action usable in any state).

- **Transition model**: how the state changes depending on the action
  deterministically $tr : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$
  probabilistically or $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$
  Simulator; forward model. deterministic or probabilistic transition.

Vocabulary 2/3

- **Rewards**: any guidance available. $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$
  How to provide rewards in simulation ? in real-life ?
  What about the robot safety ?

- **Policy**: mapping from states to actions.
  deterministic $\pi : \mathcal{S} \mapsto \mathcal{A}$ or stochastic $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$

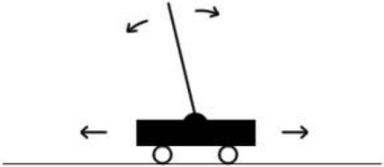
  this is the goal: finding a good policy

  good means:
  ∗reaching the goal
  ∗receiving as many rewards as possible
  ∗as early as possible.

Vocabulary 3/3

Episodic task

- Reaching a goal (playing a game, painting a car, putting something in the dishwasher)
- Do it as soon as possible
- Time horizon is finite



Single Pole Balancing

Continual task

- Reaching and keeping a state (pole balancing, car driving)
- Do it as long as you can
- Time horizon is (in principle) infinite

Case 1. Optimal control 1/2

Case 1. Optimal control 2/2

Known dynamics and target behavior

- state u, action $a \to$ new state u'
- wanted: sequence of states

Approaches

- Inverse problem
- **Optimal control**

Challenges

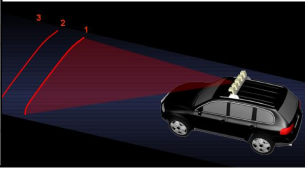
- Model errors, uncertainties
- Stability

Case 2. Reactive behaviors

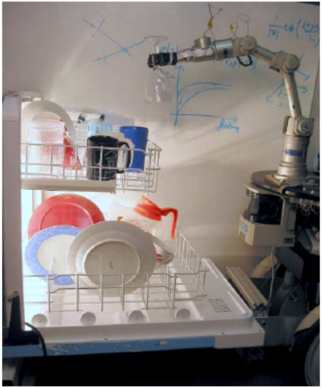
The 2005 Darpa Challenge

The terrain

The sensors



Case 3. Planning



An instance of reinforcement learning / planning problem

1. Solution = sequence of (state,action)
2. In each state, decide the appropriate action
3. ..such that in the end, you reach the goal

Case 3. Planning, 2/2

Approaches

- Reinforcement learning
- Inverse reinforcement learning
- Preference-based RL
- Direct policy search (= optimize the controller)
- Evolutionary robotics

Challenges

- Design the objective function (define the optimization problem)
- Solve the optimization problem
- Assess the validity of the solution

Games

- Backgammon: TD-Gammon 1992
- Atari 2015
- Go: AlphaGo 2016

Overview

Formalisation

Notations

- State space $\mathcal{S}$
- Action space $\mathcal{A}$
- Transition model
  - deterministic: $s' = t(s, a)$
  - probabilistic: $p(s, a, s') \in [0, 1]$.
- Reward $r(s)$ **bounded**
- Time horizon H (finite or infinite)

Goal

- Find policy (strategy) $\pi : \mathcal{S} \mapsto \mathcal{A}$
- which maximizes (discounted) cumulative reward from now to timestep H

$$\sum_t r(s_t)$$

Formalisation

Notations

- State space $\mathcal{S}$
- Action space $\mathcal{A}$
- Transition model
  - deterministic: $s' = t(s, a)$
  - probabilistic: $p(s, a, s') \in [0, 1]$.
- Reward $r(s)$ **bounded**
- Time horizon H (finite or infinite)

Goal

- Find policy (strategy) $\pi : \mathcal{S} \mapsto \mathcal{A}$
- which maximizes (discounted) cumulative reward from now to timestep H

$$\sum_{t=1}^{H} \gamma^t r(s_t) \qquad \gamma < 1$$

Formalisation

Notations

- State space $\mathcal{S}$
- Action space $\mathcal{A}$
- Transition model
  - deterministic: $s' = t(s, a)$
  - probabilistic: $p(s, a, s') \in [0, 1]$.
- Reward $r(s)$ **bounded**
- Time horizon H (finite or infinite)

Goal

- Find policy (strategy) $\pi : \mathcal{S} \mapsto \mathcal{A}$
- which maximizes (discounted) cumulative reward from now to timestep H

$$\mathbb{E}_{s_0,\pi}[\sum_{t=1}^{\infty} \gamma^t r(s_t)]$$

Markov Decision Process

But can we define $P^a_{ss'}$ and $r(s)$?

- YES, if all necessary information is in s
- NO, otherwise
  - If state is partially observable

Goal: arrive in the third branch

  - If environment (reward and transition distribution) is changing
    Reward for *first* photo of an object by the satellite

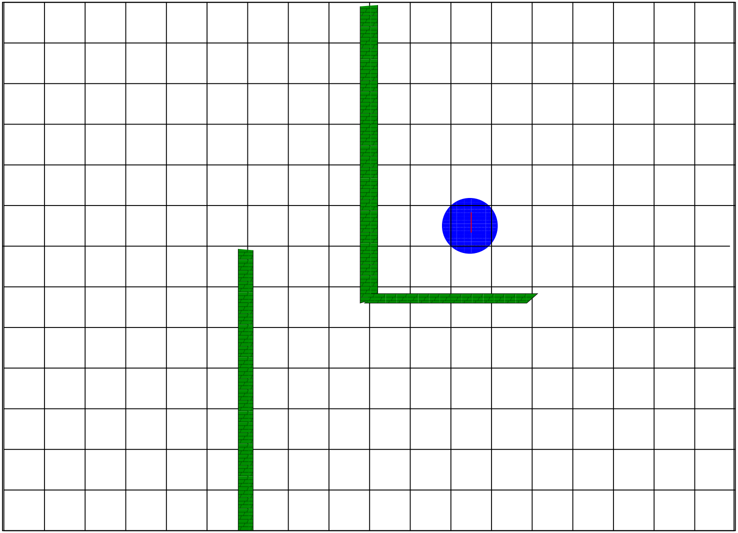
The Markov assumption

$$P(s_{h+1} | s_0\ a_0\ s_1\ a_1 \ldots s_h\ a_h) = P(s_{h+1} | s_h\ a_h)$$

Everything you need to know is the current (state, action).

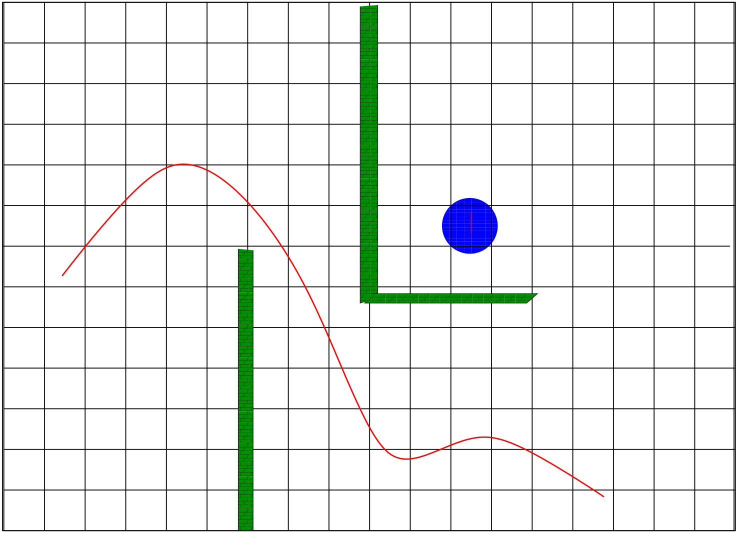
Find the treasure

Single reward: on the treasure.

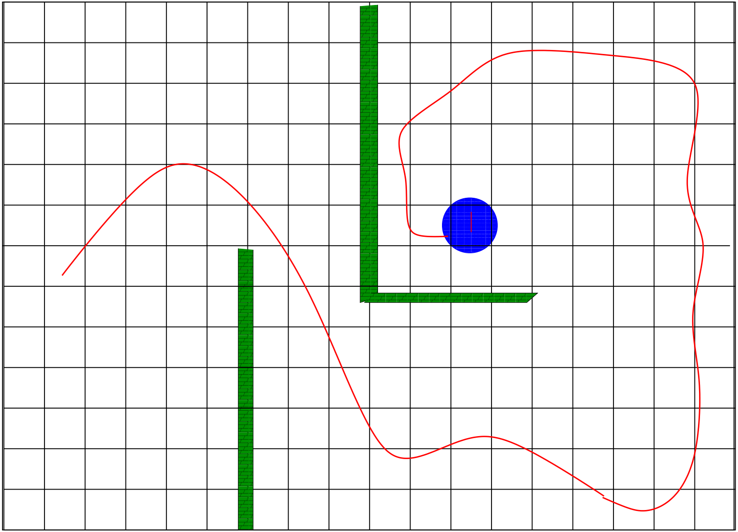


Wandering robot

Nothing happens...

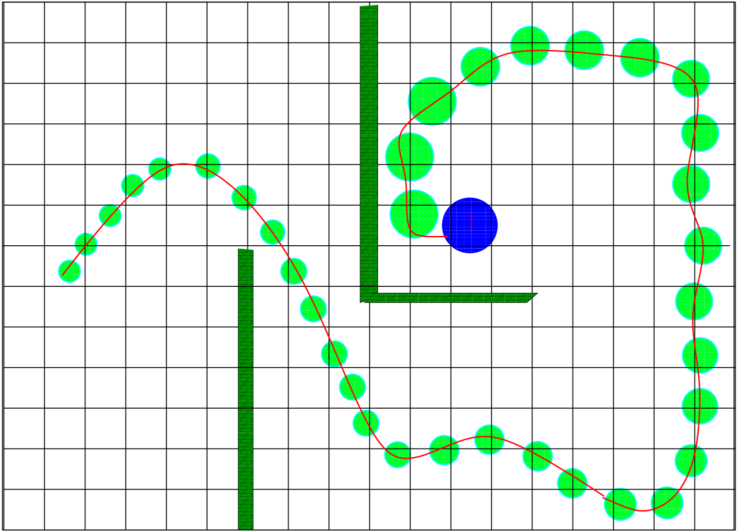


The robot finds it



Robot updates its value function

$V(s, a) ==$ "distance" to the treasure *on the trajectory*.

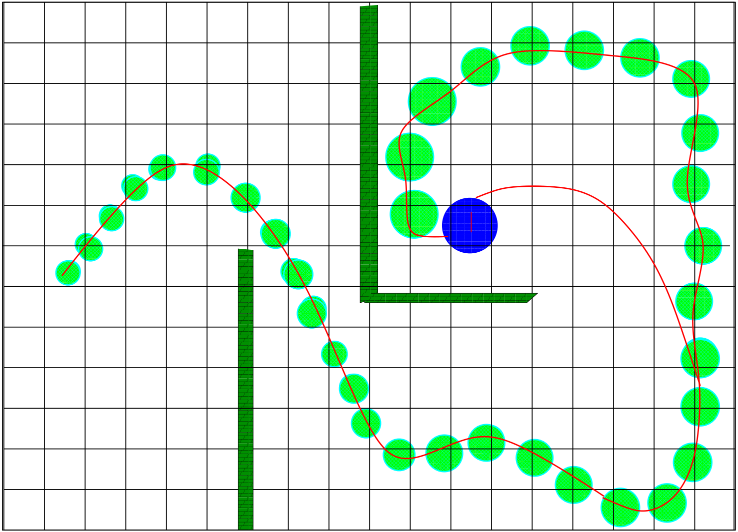


Reinforcement learning

* Robot most often selects $a = \arg\max V(s, a)$
* and sometimes explores (selects another action).

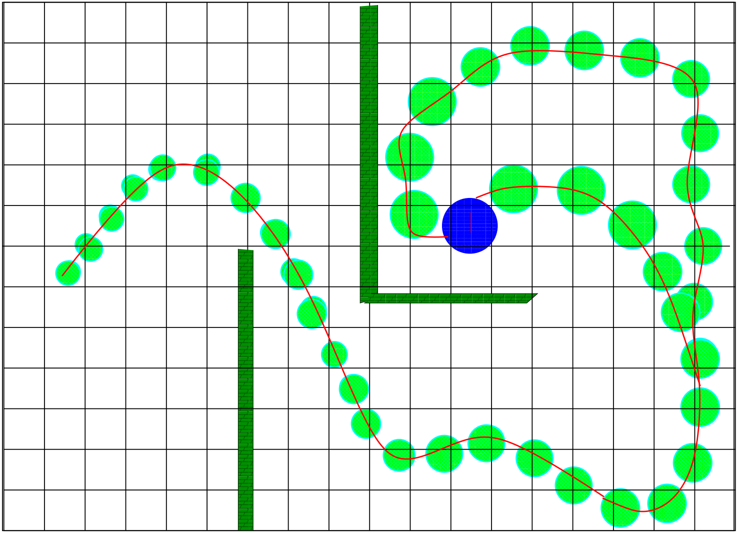
Reinforcement learning

* Robot most often selects $a = \arg\max\, V(s, a)$
* and sometimes explores (selects another action).
* Lucky exploration: finds the treasure again



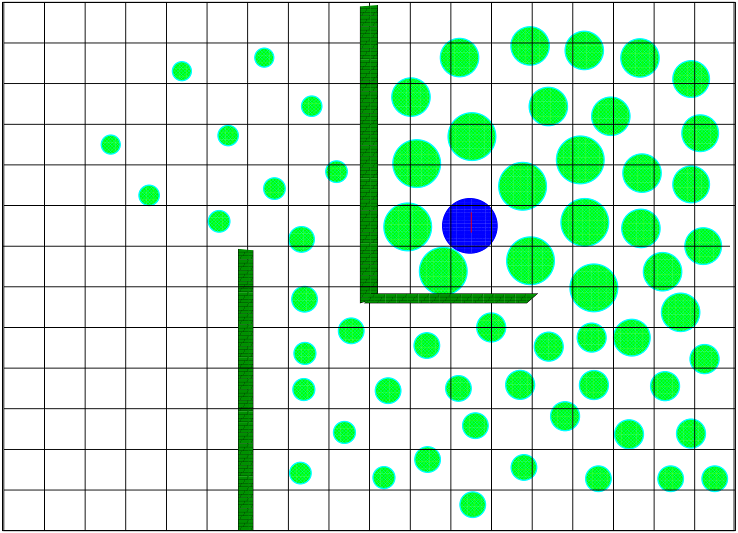
Updates the value function

* Value function tells how far you are from the treasure *given the known trajectories*.



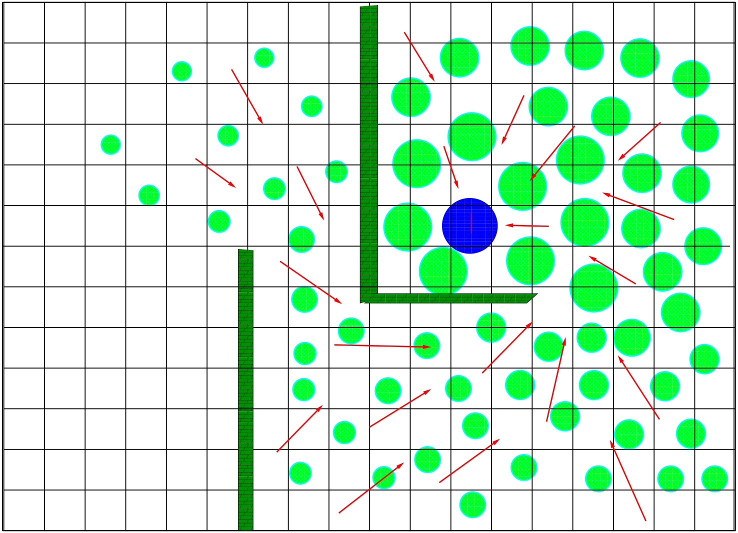
Finally

* Value function tells how far you are from the treasure



Finally

Let's be greedy: selects the action maximizing the value function



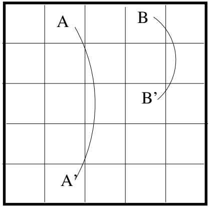
You are the learner

Rich. Sutton: https://www.microsoft.com/en-us/research/video/tutorial-introduction-to-reinforcement-learning-with-function-approximation/
15:30

Exercize

Uniform policy

- States: squares
- Actions: north, south, east, west.
- Rewards: -1 if you would get outside; 10 in A; 5 in B
- Transitions: as expected, except: A $\rightarrow$ A'; B $\rightarrow$ B'.



A –> A', reward 10

B –> B', reward 5

Compute the value function for
$\pi(s) =$ North for all s
$\gamma = .9$