Giochi e Reinforcement Learning
= GioReL su FAD

1. Introduction.
2. Dynamic Programming (DP): planning.
    1. Markov Decision Processes (MDP).
    2. Prediction, improvement, control: policy iteration.
    3. Value iteration.
3. Reinforcement Learning (RL): learning in the tabular case.
    1. Model-free prediction: Monte Carlo (MC) methods.
    2. Model-free prediction: Temporal Difference (TD) methods.
    3. Model-free control: MC methods.
    4. Model-free control: TD methods.
    5. On-policy vs off-policy methods: SARSA vs Q-learning.
4. Multi-armed bandit. Very likely.
5. MCTS. Likely.
6. Reinforcement Learning (RL): learning in the function approximation case. Maybe.

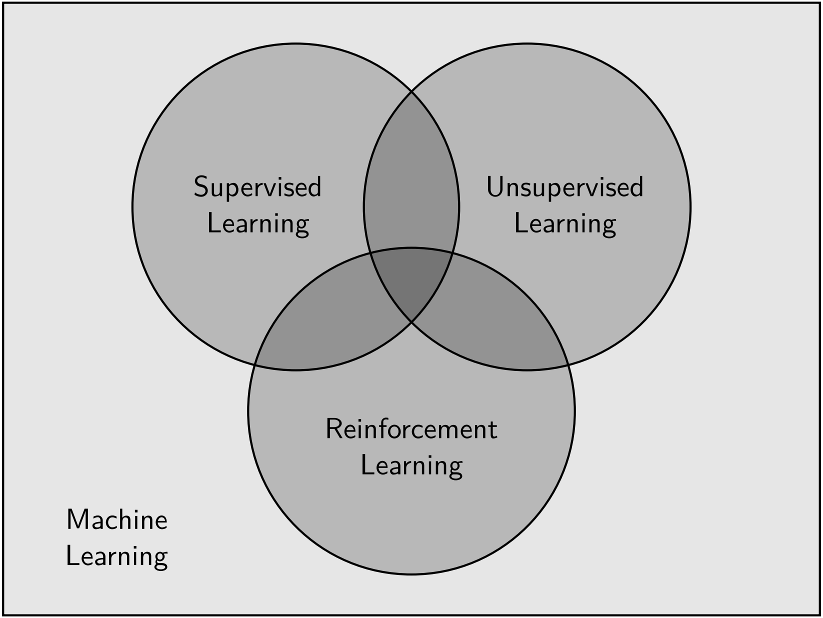
Acknowledgements, sources and links

Both the organization and the content of the slides are extracted from the following sources:

- Reinforcement Learning: An Introduction. Richard S. Sutton and Andrew G. Barto, second edition, 2018.
- UCL Course on RL, videos and slides. David Silver, 2015.
- Tutorial: Introduction to Reinforcement Learning with Function Approximation. Richard S. Sutton, 2016.
- Implementation of Reinforcement Learning algorithms. Denny Britz, GitHub project, 2016 (updated in 2018).

Introduction: Who, What, When, Where, Why, hoW

RL is not SL, RL is not UL



RL characteristics

What is RL?

- Agent-oriented learning: an *agent* learns by *interacting with an environment* to achieve a *goal*.
- The agent learns by *trial and error*, evaluating a delayed feedback (*reward*).
- The kind of machine learning most like natural learning.
- Learning that can tell for itself when it is right or wrong.

RL vs SL and UL

- RL is not completely supervised: only reward.
- RL is not completely unsupervised: there is reward.
- *Time* matters: *sequential data*.
- Time matters: actions change possible future.

Examples

Real world applications of RL (original article)

- Resources management in computer clusters.
- Traffic light control.
- Robotics.
- Web system configuration.
- Chemistry.
- Personalized recommendations.
- Bidding and advertising.

Examples

More specific tasks with their goal

- Adaptive controller: adjusts parameters of a petroleum refinery's operation in real time.
- Gazelle calf. Learn to run.
- Trash-collecting mobile robot. Collect trash.
- Preparing breakfast. Feed yourself.
- Chess player. Win (or enjoy).

Examples

Games

- AlphaGo's family.
- StarCraft II. Very recent achievement, 19 Dec 2018.
- Atari games. Very recent achievement, 28 Sep 2018.
- TD-Gammon.

Enjoy few minutes of video

- Atari:
  https://www.youtube.com/watch?v=V1eYniJ0Rnk&vl=en
- AlphaGo:
  https://www.youtube.com/watch?v=8dMFJpEGNLQ
- StarCraft: https://youtu.be/UuhECwm31dM

1. What is Reinforcement Learning?
2. Examples
3. The RL setup: problem and actors
4. What do we know? State, observability and distribution model
5. What can we do? Policy and value
6. The never-ending control loop: prediction $\rightleftharpoons$ improvement
7. Planning, learning and the XX compromise

The RL problem

Common points in examples

- Trying to reach a *goal*.
- Interactions: active *decision-making agent vs environment*.
- *Uncertainty* about the environment.
- Effects of actions cannot be fully predicted: adaptation required (*learning*).

The RL reward hypothesis

All goals can be described by the maximization of expected cumulative reward (the *value*).

- Is it true? Interesting analysis at http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html.
- Related with the *expected utility hypothesis* from von Neumann-Morgenstern utility theory.

The RL problem

RL main task

Decision problem: we would like to choose actions that maximize the *return*, i.e. the total future reward.

Sequential decision making

Actions may have long term consequences.

Uncertainty

The best we can aim for is maximizing the *value*, i.e. the *expected* total future reward.

Exercise

Find an example of a *deterministic* task, that is, a task where you know the outcome of your actions.

The RL problem

To be *greedy* can be wrong

- A financial investment (may take months to mature).
- Refuelling a helicopter (might prevent a crash in several hours).
- Blocking opponent moves (might help winning chances many moves from now).

Exercise

Discuss the difference between return and value.

The RL problem

Examples of reward

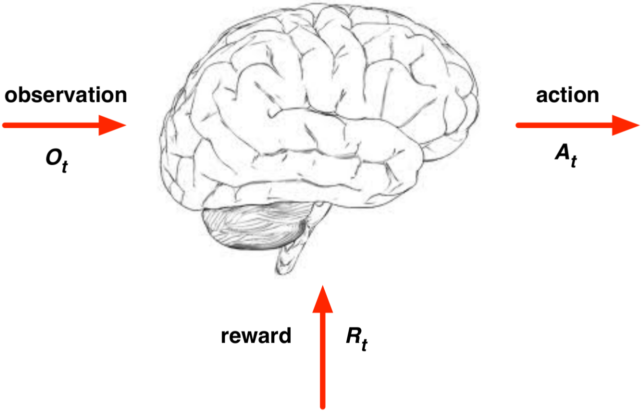
- Games: $R_T := -1, 0, +1$ (win, draw, lose). More generally, R_T can be the final score.
- Games: $R_T := 0, +1$ (win, lose). In this case, the value is the probability of winning. Why?
- Atari games: R_t is the immediate score increment at step t.
- Walking robot: $R_t := +1$ for every step he doesn't fall. https://www.youtube.com/watch?v=gn4nRCC9TwQ
- Financial investment: R_t is the money increment in the last time step in portfolio.
- Maze and Gridworld: $+100$ for reaching the exit, 0 otherwise. Wrong. Why?

The RL problem

Examples of reward

- Games: $R_T := -1, 0, +1$ (win, draw, lose). More generally, R_T can be the final score.
- Games: $R_T := 0, +1$ (win, lose). In this case, the value is the probability of winning. Why?
- Atari games: R_t is the immediate score increment at step t.
- Walking robot: $R_t := +1$ for every step he doesn't fall. https://www.youtube.com/watch?v=gn4nRCC9TwQ
- Financial investment: R_t is the money increment in the last time step in portfolio.
- ~~Maze and Gridworld: $+100$ for reaching the exit, 0 otherwise. Wrong. Why?~~
- Maze and Gridworld: -1 for every move. Correct. Why?

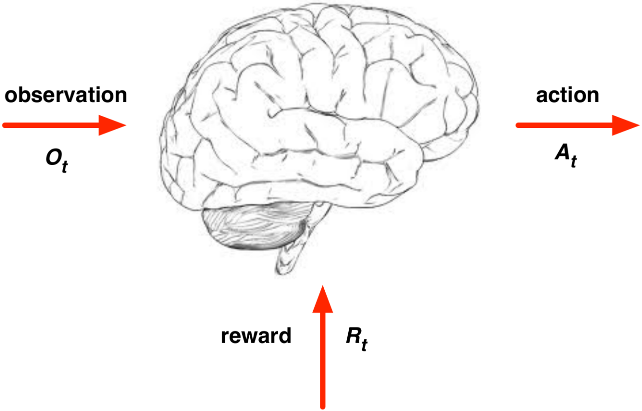
First actor: the *agent*



A never-ending loop

- …we (the agent) receive R_t and observe O_t…
- …we choose the action $A_t \sim \pi(\cdot, f(O_t, R_t, A_{t-1}, O_{t-1}, R_{t-1}, \dots))$…
- …and because of our action A_t, the environment send us a reward R_{t+1} and a new *state*, that we observe as O_{t+1}…

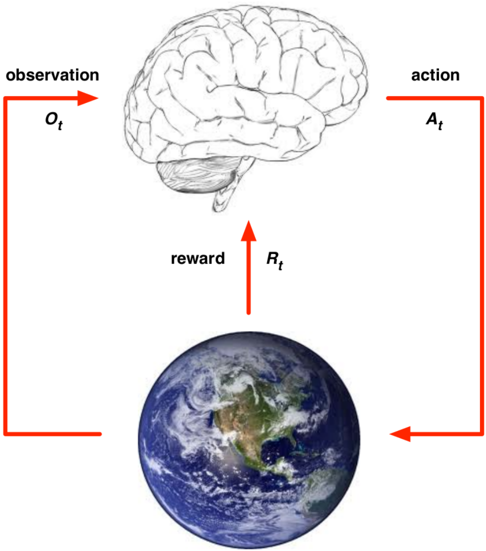
First actor: the *agent*



A never-ending loop

- . . . we (the agent) receive R_t and observe O_t. . .
- . . . we choose the action $A_t \sim \pi(\cdot, f(\mathit{history}))$. . .
- . . . and because of our action A_t, the environment send us a reward R_{t+1} and a new *state*, that we observe as O_{t+1}. . .

We are not alone! Second actor: the *environment*



Agent, step t

- Receives observation O_t.
- Receives scalar reward R_t.
- Computes his own *state* S_t^a.
- Executes action A_t.

Environment, step t

- Receives action A_t.
- Computes his own *state* S_{t+1}^e.
- Emits observation O_{t+1}.
- Emits scalar reward R_{t+1}.

History, *agent state* and *environment state*

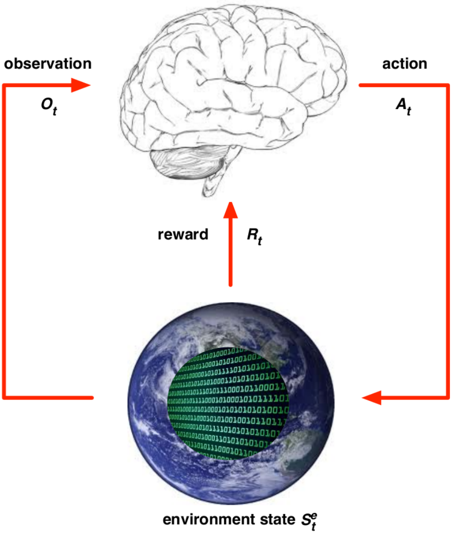
Notation

- *History*: the sequence of observations, actions, rewards up to time step t:

$$H_t := O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t.$$

- The agent selects actions, and the environment answers with *observations* and *rewards*.
- *State*: the information used (by the agent and the environment) to determine what happens next.
- State is naturally a sequence S_t.
- Agent state is a function of history: $S_t := f(H_t)$.
- *Environment state* S_t^e is different from *agent state* S_t^a.

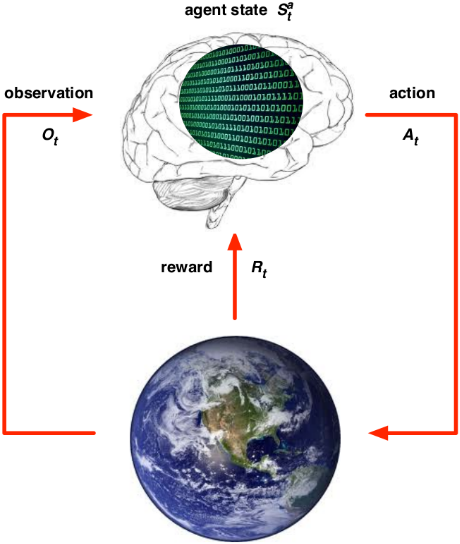
Environment state



Environment, step t

- Environment state S_t^e: data the environment uses to pick the next observation and reward.
- S_t^e is not usually visible to the agent.
- Even if S_t^e is visible, it may contain irrelevant information.

Agent state



Agent, step t

- Agent state S_t^a: data the agent uses to pick the next action.
- S_t^a is the information used by RL algorithms.
- S_t^a can be any function of history: $S_t^a := f(H_t)$.

Markov state

Uncertainty

Since we have no control of environment, everything is a *random variable*.

Definition

A sequence of states (random variables) is *Markov* if and only if

$$\Pr(S_{t+1}|S_t) = \Pr(S_{t+1}|S_1, \ldots, S_t)$$

- The future is independent of the past given the present:

$$S_t \rightarrow H_{t+1:+\infty}$$

- Once the state is known, the history may be thrown away: the state is a sufficient statistic of the future.

Exercise

Is the environment state S_t^e Markov? Is the history H_t Markov?

Partially observable environments

The agent indirectly observes environment.

- Robot with camera vision, no absolute location: $O_t =$ camera image at time t.
- Poker playing agent: $O_t =$ public cards at time t.

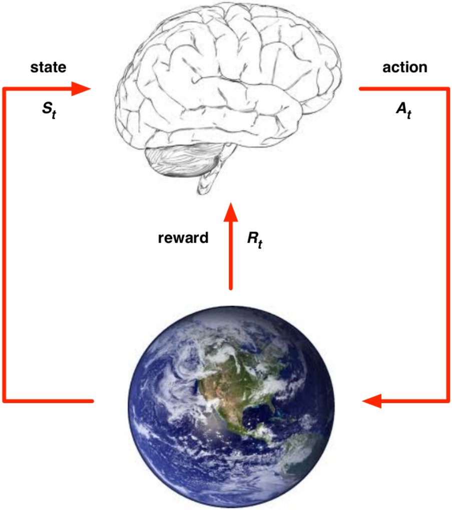
Agent must construct its own state representation S_t^a. For instance:

- Complete history: $S_t^a := H_t$.
- Beliefs of environment state:

$$S_t^a := (P(S_t^e = s_1), \ldots, P(S_t^e = s_n)).$$

- Recurrent neural network approximation: $S_t^a := \sigma(S_{t-1}^a W_s + O_t W_o)$.

Fully observable environments



- The agent directly observes environment state: $O_t = S_t^a = S_t^e$.
- Agent state and environment state coincides!

A never-ending loop

- ... we (the agent) receive R_t and observe S_t...
- ... and thus we decide to do action $A_t \sim \pi(\cdot, S_t)$...
- ... and because of our action A_t, the environment send us a reward R_{t+1} and a new state, that we observe as S_{t+1}...

Structure of an RL environment

Distribution model

- Predicts what the environment will do next, via a probability distribution p that predicts the next state and reward:

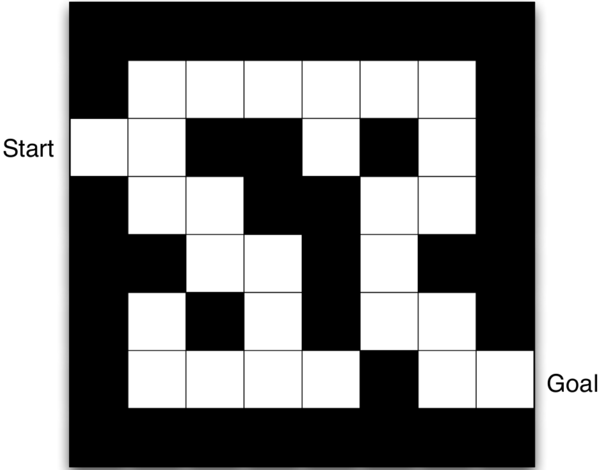
$$p(s', r|s, a) := \Pr(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a).$$

- If we have p, we can predict next state and next reward, we can compute the average next reward, and so on.

Remarks

- We assume the Markov property. Exercise: write it.
- Usually, we don't know p. For this reason it is called *model*.
- Model: our representation of the environment. Can be perfect (a game with rules) or not (weather forecast).
- We assume that the environment is *time homogeneous*: p does not depend on t. Exercise: is this usually true?

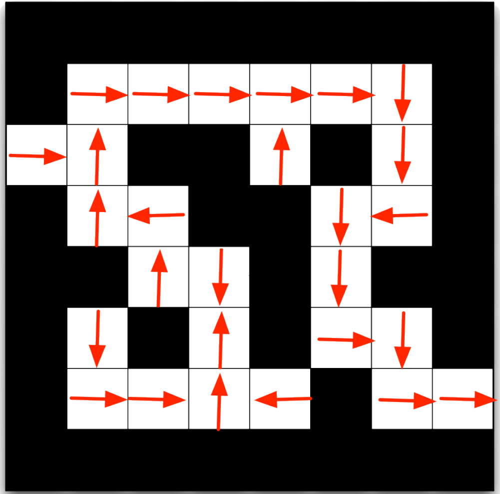
Example: the maze



Exercise

Discuss this example in terms of the language you have learned up to now.

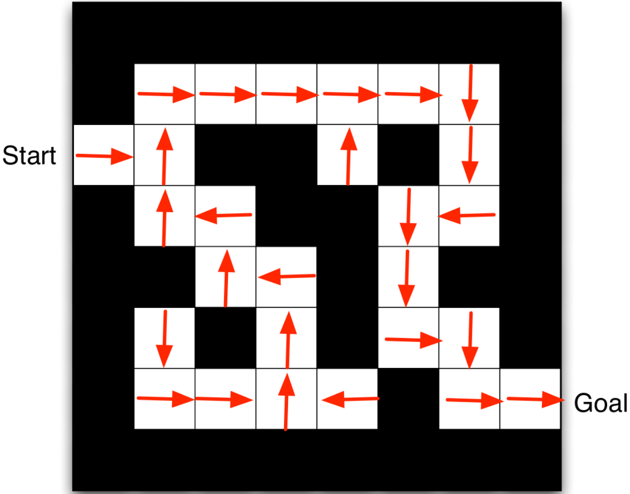
Example: a *policy* for the maze



~~Strategy~~Policy

Arrows represent the *policy* π: which action to take from every state.

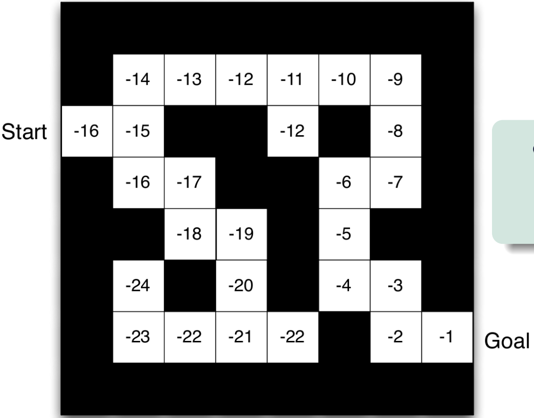
Example: *optimal policy* for the maze



Exercise

Choose a state s (any state, not only start) and *follow the policy*. Would you call this policy *optimal*?

Example: values of the optimal policy for the maze



- Value $v_\pi(s)$ for every state s, for the optimal policy π of previous slide.

Exercise

Choose a state s and compute $v_\pi(s)$ by yourself. If s' denote the successor state of s, can the value $v_\pi(s')$ help with this computation?

Prediction, *improvement* and *control*

The *prediction* problem in RL

Forecast the future: can you say from each state how much will be your return? It depends on the policy!

The *improvement* problem in RL

Change the future: can you find a different policy that will give you a better return?

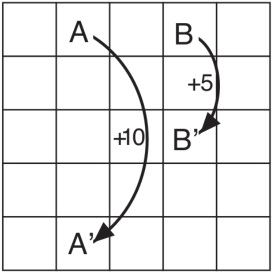
The *control* problem in RL

Change the future: can you find the best policy at all?

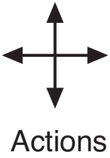
Exercise

State formally the prediction, the improvement and the control problem.

Gridworld example: prediction



(a)



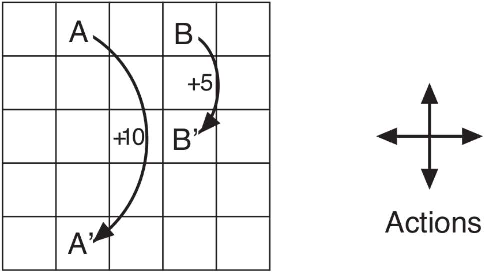
3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(b)

Exercise

Compute the value function for the uniform random policy.

Gridworld example: improvement



(a)

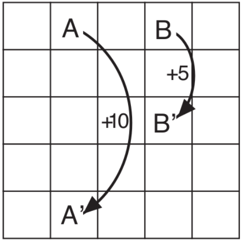
3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

(b)

Exercise

Find an improvement of the uniform policy.

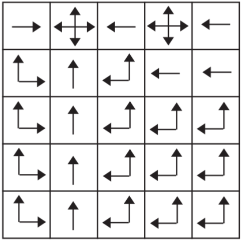
Gridworld example: *optimal control*



a) gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

b) v_*



c) π_*

Exercise

- Compute the *optimal value* function over all possible policies.
- Given the optimal value v_* as above, find the optimal policy.
- Is the optimal policy unique?

Two ways to solve the RL problem: *planning* and *learning*

Two fundamental problems appear in sequential decision making: *planning* and *learning*.

Planning (*dynamic programming*)

- A distribution model of the environment is known.
- The agent performs computations via the distribution model, no external interaction with the environment. Average return.
- The agent improves its policy.

Learning (*reinforcement learning*)

- The environment is initially unknown.
- The agent interacts with the environment, hopefully via a *sample model*. Empirical mean of return.
- The agent improves its policy.

Planning and learning: similarities

Look ahead

Both planning and learning are based on looking ahead to future events, computing a backed-up value, and then using it as an update target for an approximate value function.

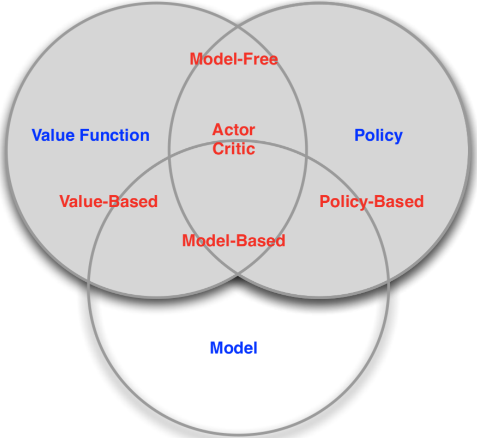
Value functions evaluation

The heart of both planning and learning is the computation of value functions for states and actions.

Policy improvement

The heart of both planning and learning is the improvement of the policy.

Categorizing RL agents



- Value based: no policy (implicit), value function.
- Policy based: policy, no value function.
- Actor-Critic: policy, value function used to improve the policy.

- Model based: policy and/or value function, model.
- Model free: policy and/or value function, no model.

Exercise for the future

Put in this taxonomy the RL algorithms you will learn.

Exploration vs exploitation: the eternal dilemma

Old and certain, or new but unsure?

- Reinforcement learning is trial-and-error learning.
- Take actions that usually give high reward? Exploitation!
- Take actions that were never explored? Exploration!
- The policy should make a compromise between exploration and exploitation.

Examples

- Restaurant selection: favourite place or new try?
- Oil drilling: best location or promising spot?
- Game playing: best or experimental move?

Exercise: multi-armed bandit

Suppose you have 10 *different* slot machines where you can play. You have 1000€, each play costs 1€. Propose an exploration/exploitation policy for maximizing your final return.

Wrapping up

Learning goals

- Understand the RL problem, and how RL differs from supervised learning.
- Understand reward, return and how they are used to make decisions.
- Understand actions, states and rewards in term of agent/environment interactions.

Wrapping up

What we (hopefully) have learnt

- Reinforcement Learning (RL) is concerned with goal-directed learning and decision-making.
- In RL an agent learns from experiences it gains by interacting with the environment. In supervised learning we cannot affect the environment.
- In RL rewards are often delayed in time and the agent tries to maximize a long-term goal. For example, one may need to make seemingly suboptimal moves to reach a winning position in a game.
- An agent interacts with the environment via actions. The environment answers with states and rewards.