Reinforcement learning in neuroscience - I

Rita Almeida

PhD course 3045, VT 2018

Reinforcement learning

- Goal-directed learning from interactions.
- Learning by trial-and-error what to do in a given situation in order to maximize total reward and minimize total punishment.

- trial-and-error
- sense the situation
- action
- goals
- Difficulties:
  - reward and punishment can be delayed
  - outcomes may depend on a series of actions

Reinforcement learning in different fields

Computer science

Artificial intelligence

Neuroscience

Dopamine
Reward system
...

Reinforcement learning

Optimal control

Engineering

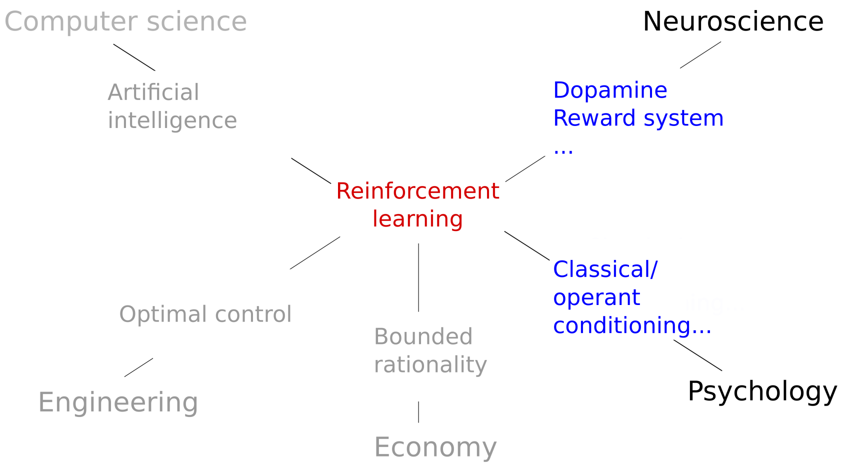
Bounded rationality

Economy

Classical/ operant conditioning...

Psychology

RL in neuroscience and psychology



Learning

Neuroscience of learning:

- Mechanisms:
  - Activity dependent synaptic plasticity
  - Long term potentiation and depression (LTP, LTD)
  - Hebbian learning
  - Dopamine
  - Acetylcholine...
- Structures and circuits:
  - Basal ganglia...
- Cognitive neuroscience / psychology:
  - Implicit versus explicit
  - Associative versus non-associative
  - Classical and operant conditioning...

Learning

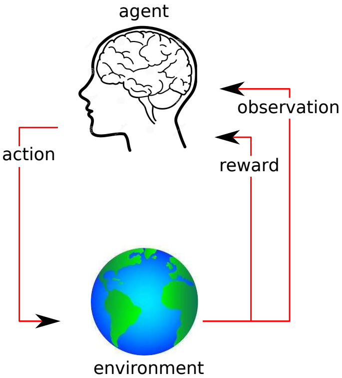
Computational approaches to learning:

- Machine / statistical learning:
  - supervised learning - learning from examples from a knowledgeable external supervisor.
  - unsupervised learning - learning from data.
- Reinforcement learning - agent learning from interactions, from own experience.
  - Suitable for problems where it is impractical to have examples of all situations where the agent has to act.
  - Feedback can be delayed.
  - Data is acquired in a sequence.
  - Agent's actions affect the data it receives.

Key aspects of RL

Learning problem where an agent interacts with the environment to achieve a goal.

- Sensation: observe the state of the environment
- Action: Take actions that affect the state of the environment
- Goal: Relating to state of environment and reward



Examples

- Animal learning to get food.
- Robot learning to escape a maze.
- Financial investment.
- Learning to play Backgammon.
- Control an industrial machine to keep a given temperature.

Key aspects of RL

Learning requires trade-off between:

- Exploitation - agent prefers actions that were effective before.
- Exploration - agent explores actions in order to make better future selections. Assures that:
  - the agent does not get stuck with a good but not optimal action.
  - the expected reward is properly estimated in a stochastic task.
  - the agent adapts when the task is non-stationary.

Elements of RL

Policy: mapping between perceived states and actions.

- Can be stochastic.

Reward function: mapping between state (or state-action) to a value summarizing the desirability of that state.

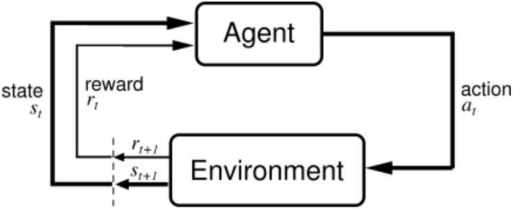
- Agent wants to maximize the total reward.
- Directly given by environment.
- Can be stochastic.

Value function: mapping between state to the long term desirability of that state.

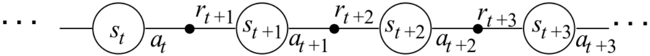
- Takes into account total amount of reward the agent expects to accumulate from that state on.
- Action choices are made based on value.
- Value is estimated from all observations the agent does.

Model of the environment: (optional) a model of the behavior of the environment, used for planning.

Defining the RL problem



- Agent and environment interact at a sequence of time steps $t = 0, 1, 2, 3, \ldots$
- Agent observes at step t state $s_t \in \mathcal{S}$
- Agent produces at step t action $a_t \in \mathcal{A}$
- Agent gets resulting reward $r_{t+1} \in \mathcal{R}$
- Agent gets into the resulting state s_{t+1}



Figures adapted from Sutton and Barto, 1998.

Defining the RL problem - policy

Policy at step t, π_t: mapping from states to probabilities of selecting each possible action.

$\pi_t(a, s)$ is the probability of $a_t = a$ if $s_t = s$

- RL specifies how the agent changes policy based on experience.
- The goal of the agent is to maximize the reward it receives in the long run.

Defining the RL problem - goals and rewards

- Rewards must be defined in a way that maximizing them corresponds to achieving the goal.
- Examples:
  - Animal learning to select a box to find food: reward +1 when it finds food.
  - Learning to escape a maze: reward -1 for each step and +1 to escape.
  - Financial investment: reward proportional to the money gained and lost.
  - Learning to play chess: reward + / - to winning / losing the game
    - Not rewarding for each piece taken!

Defining the RL problem - returns

Return: sum of future rewards.

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T$$

Discounted return, with the discount rate γ:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \qquad 0 \leq \gamma \leq 1$$

- Does not need a final time step.

Markov decision processes (MDP)

- MDPs are reinforcement learning tasks that satisfy the Markov property (assuming finite number of states and reward values):

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\}$$

- If the sate and action spaces are finite - finite Markov decision processes. Defined by:
  - Transition probabilities: $\mathcal{P}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$
  - Expected reward: $\mathcal{R}^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s\}$

Defining the RL problem - value functions

State-value function (for policy π) estimates how desirable it is for an agent to be in a given state. For MDPs:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\}$$

Action-value functions (for policy π) estimates how desirable it is for an agent to perform a given action in a given state.

$$Q^{\pi}(s, a) = E_{\pi}\{R_t | s_t = s, a_t = a\}$$

- V^{π} and Q^{π} can be estimated from experience.

Bellman equation for V_π

- Value of a state can be written as a function of values of successor states - recursive relationship.

$$\begin{aligned} V^\pi(s) &= E_\pi\{R_t | s_t = s\} = E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} \\ &= E_\pi\{\gamma^0 r_{t+0+1} + \gamma(\gamma^0 r_{t+1+1} + \gamma^1 r_{t+2+1} + \dots) | s_t = s\} \\ &= E_\pi\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\} \\ &= \dots \end{aligned}$$

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V^\pi(s')]$$

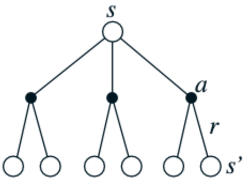
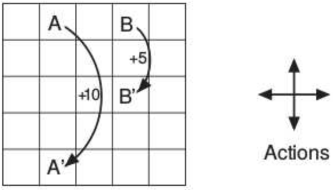


Figure adapted from Sutton and Barto, 1998.

Example - gridworld

- Agent can move 1 cell at a time north, south, east and west.
- Actions that take the agent off the grid result in reward -1 and not moving.
- Reaching A (B) moves the agent to A' (B') and gives reward +10 (+5).
- All other states reward 0.
- Policy: selecting each direction with equal probability.
- Discount rate $\gamma = 0.9$



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

Figure adapted from Sutton and Barto, 1998.

- Expected returns for each position are indicated - state-value function.
- For A expected return less that reward and for B larger.

Optimal policies and value functions

- To solve a RL task one wants to find a policy that achieves a lot of reward in the long run.
- Policy π' is better than π if $V^{\pi'}(s) \geq V^{\pi}(s)$ for all s.

Optimal policies π^* have optimal state-value function:

$$V^*(s) = max_\pi V^\pi(s)$$

Bellman optimality equation:

$$V^*(s) = max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^*(s')]$$

Example - gridworld

Optimal Policy and state-value function:

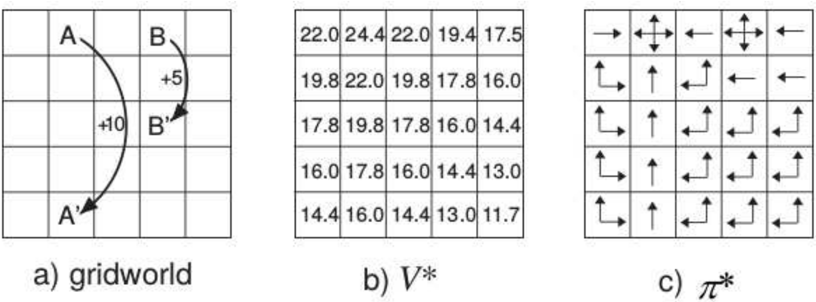


Figure adapted from Sutton and Barto, 1998.

Finding an optimal policy

- Solving the optimality Bellman's equations is many times not feasible.
  - problem has to be MDP
  - environment dynamics have to be known
  - computation time too long
  - memory too little
- Approximations are used.
- Many RL methods can be seen as approximations to solving the Bellman optimality equation.

Bibliography