University of Rome "La Sapienza"

Master in Artificial Intelligence and Robotics

Machine Learning

A.Y. 2018/2019

Prof. Luca Iocchi

Sapienza University of Rome, Italy
Master in Artificial Intelligence and Robotics
Machine Learning (2018/19)

9. Markov Decision Processes and Reinforcement Learning

Luca Iocchi

Overview

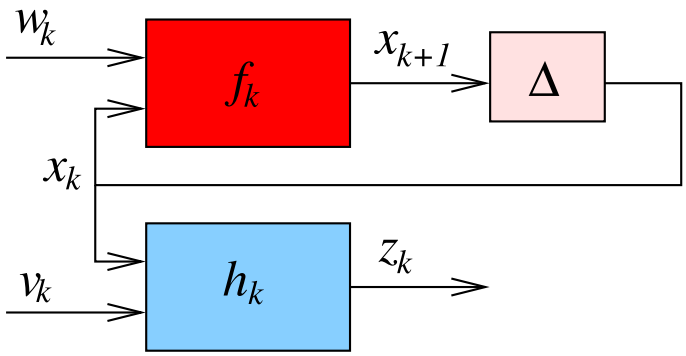
- Dynamic systems
- Markov Decision Processes
- Q-Learning
- Experimentation strategies
- Evaluation

References

Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. (2nd edition).
On-line: http://incompleteideas.net/book/the-book.html

Dynamic System

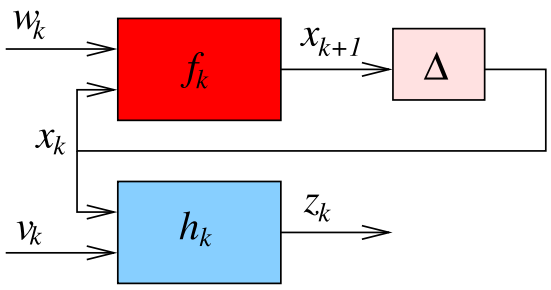
The classical view of a dynamic system



x: state
z: observations
w, v: noise

f: state transition model
h: observation model

Reasoning vs. Learning in Dynamic Systems



Reasoning: given the model (f, h) and the current state x_k, predict the future (x_{k+T}, z_{k+T}).

Learning: given past experience ($z_{0:k}$), determine the model (f, h).

State of a Dynamic System

The state x encodes:

- all the past knowledge needed to predict the future.
- the knowledge gathered through operation
- the knowledge needed to pursue the goal

Examples:

- configuration of a board game
- configuration of robot devices
- screenshot of a video-game

Observability of the state

When the state is fully observable, the decision making problem for an agent is to decide which *action* must be executed in a given *state*.

The agent has to compute the function

$$\pi : \mathbf{X} \mapsto A$$

When the model of the system is not known, the agent has to *learn* the function π.

Supervised vs. Reinforcement Learning

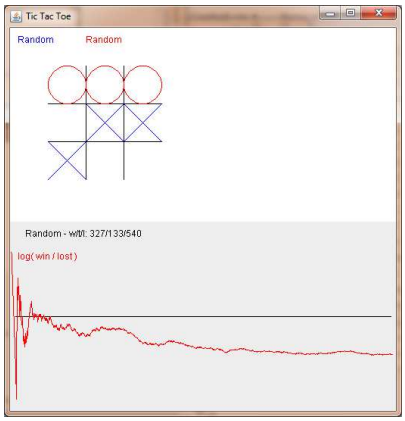
Supervised Learning
Learning a function $f : X \rightarrow Y$, given

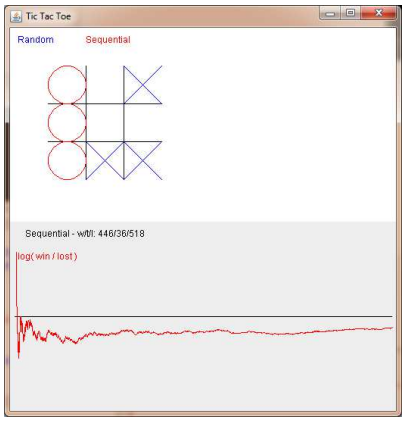
$$D = \{\langle x_i, y_i \rangle\}$$

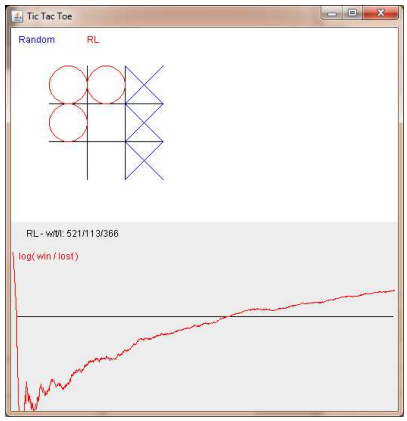
Reinforcement Learning
Learning a behavior function $\pi : \mathbf{X} \rightarrow A$, given

$$D = \{\langle \mathbf{x}_1, a_1, r_1, \ldots, \mathbf{x}_n, a_n, r_n \rangle^{(i)}\}$$

RL Example: Tic Tac Toe





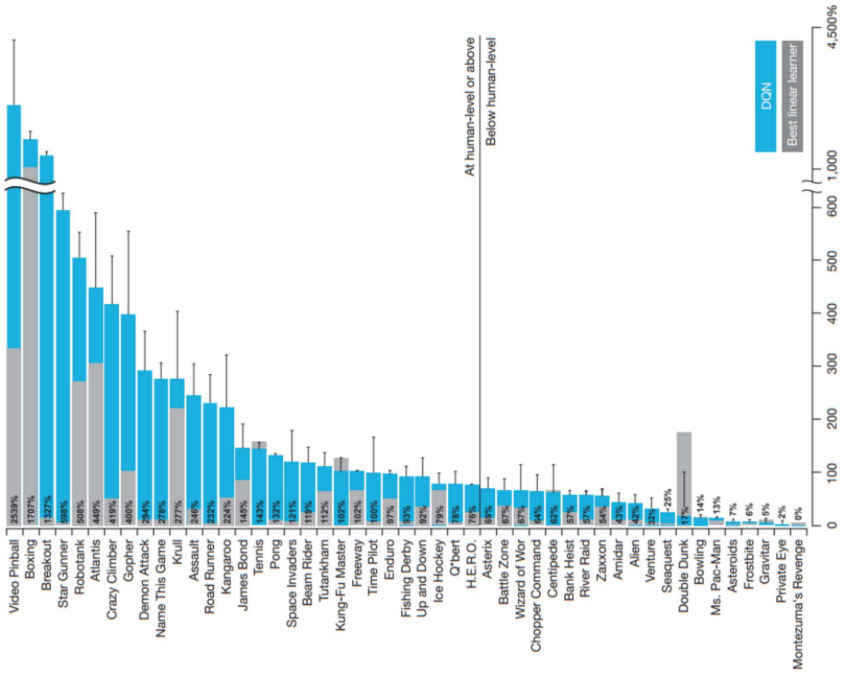


RL Example: Humanoid Walk

RL Example: Controlling an Helicopter

Deep Reinforcement Learning: ATARI games

Deep Q-Network (DQN) by Deep Mind (Google)



Human-level control through Deep Reinforcement Learning, Nature (2015)

Dynamic System Representation

X: set of states

- explicit discrete and finite representation $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$
- continuous representation $\mathbf{X} = F(\ldots)$ (state function)
- probabilistic representation $P(\mathbf{X})$ (probabilistic state function)

A: set of actions

- explicit discrete and finite representation $\mathbf{A} = \{a_1, \ldots, a_m\}$
- continuous representation $\mathbf{A} = U(\ldots)$ (control function)

δ: transition function

- deterministic / non-deterministic / probabilistic

Z: set of observations

- explicit discrete and finite representation $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$
- continuous representation $\mathbf{Z} = Z(\ldots)$ (observation function)
- probabilistic representation $P(\mathbf{Z})$ (probabilistic observation function)

Markov property

Markov property

- Once the current state is known, the evolution of the dynamic system does not depend on the history of states, actions and observations.
- The current state contains all the information needed to predict the future.
- Future states are conditionally independent of past states and past observations given the current state.
- The knowledge about the current state makes past, present and future observations statistically independent.

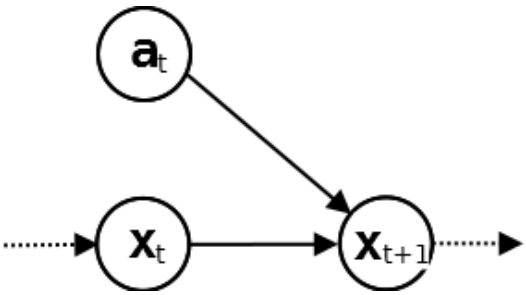
Markov process is a process that has the Markov property.

Markov Decision Processes (MDP)

Markov processes for decision making.

States are fully observable, no need of observations.

Graphical model



Markov Decision Processes (MDP)

Deterministic transitions

$$MDP = \langle \mathbf{X}, \mathbf{A}, \delta, r \rangle$$

- $\mathbf{X}$ is a finite set of states
- $\mathbf{A}$ is a finite set of actions
- $\delta : \mathbf{X} \times \mathbf{A} \rightarrow \mathbf{X}$ is a transition function
- $r : \mathbf{X} \times \mathbf{A} \rightarrow \Re$ is a reward function

Markov property: $\mathbf{x}_{t+1} = \delta(\mathbf{x}_t, a_t)$ and $r_t = r(\mathbf{x}_t, a_t)$
Sometimes, the reward function is defined as $r : \mathbf{X} \rightarrow \Re$

Markov Decision Processes (MDP)

Non-deterministic transitions

$$MDP = \langle \mathbf{X}, \mathbf{A}, \delta, r \rangle$$

- $\mathbf{X}$ is a finite set of states
- $\mathbf{A}$ is a finite set of actions
- $\delta : \mathbf{X} \times \mathbf{A} \rightarrow 2^{\mathbf{X}}$ is a transition function
- $r : \mathbf{X} \times \mathbf{A} \times \mathbf{X} \rightarrow \Re$ is a reward function

Markov Decision Processes (MDP)

Stochastic transitions

$$MDP = \langle \mathbf{X}, \mathbf{A}, \delta, r \rangle$$

- $\mathbf{X}$ is a finite set of states
- $\mathbf{A}$ is a finite set of actions
- $P(\mathbf{x}'|\mathbf{x}, \mathbf{a})$ is a probability distribution over transitions
- $r : \mathbf{X} \times \mathbf{A} \times \mathbf{X} \rightarrow \Re$ is a reward function

Full Observability in MDP

States are fully observable.

In presence of non-deterministic or stochastic actions, the state resulting from the execution of an action is not known *before* the execution of the action, but it can be fully observed *after* its execution.

MDP Solution Concept

Given an MDP, we want to find an *optimal policy*.

Policy is a function

$$\pi : \mathbf{X} \rightarrow \mathbf{A}$$

For each state $\mathbf{x} \in \mathbf{X}$, $\pi(\mathbf{x}) \in \mathbf{A}$ is the *optimal* action to be executed in such state.

optimality = maximize the cumulative reward.

Optimality is defined with respect to maximizing the (expected value of the) *cumulative discounted reward*.

$$V^{\pi}(\mathbf{x}_1) \equiv E[\bar{r}_1 + \gamma \bar{r}_2 + \gamma^2 \bar{r}_3 + \ldots]$$

where $\bar{r}_t = r(\mathbf{x}_t, a_t, \mathbf{x}_{t+1})$, $a_t = \pi(\mathbf{x}_t)$, and $\gamma \in [0, 1]$ is the discount factor for future rewards.

Optimal policy: $\pi^* \equiv \text{argmax}_{\pi}\, V^{\pi}(\mathbf{x})\,,\ \forall \mathbf{x} \in \mathbf{X}$

Value function

Deterministic case

$$V^{\pi}(\mathbf{x}) \equiv r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots$$

Non-deterministic/stochastic case:

$$V^{\pi}(\mathbf{x}) \equiv E[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots]$$

Optimal policy

π^* is an **optimal policy** iff for any other policy π

$$V^{\pi^*}(\mathbf{x}) \geq V^{\pi}(\mathbf{x}), \forall \mathbf{x}$$

For infinite horizon problems, a stationary MDP always has an optimal stationary policy.

Reasoning and Learning in MDP

Problem: MDP $\langle \mathbf{X}, \mathbf{A}, \delta, r \rangle$

Solution: Policy $\pi : \mathbf{X} \rightarrow \mathbf{A}$

If the MDP $\langle \mathbf{X}, \mathbf{A}, \delta, r \rangle$ is completely known $\rightarrow$ reasoning or planning

If the MDP $\langle \mathbf{X}, \mathbf{A}, \delta, r \rangle$ is not completely known $\rightarrow$ learning

Note: Simple examples of reasoning in MDP can be modeled as a search problem and solved using standard search algorithm (e.g., A*).

One-state Markov Decision Processes (MDP)

$$MDP = \langle \{\mathbf{x}_0\}, \mathbf{A}, \delta, r \rangle$$

- $\mathbf{x}_0$ unique state
- $\mathbf{A}$ finite set of actions
- $\delta(\mathbf{x}_0, a_i) = \mathbf{x}_0, \ \forall a_i \in \mathbf{A}$ transition function
- $r(\mathbf{x}_0, a_i, \mathbf{x}_0) = r(a_i)$ reward function

Optimal policy: $\pi^*(\mathbf{x}_0) = a_i$

Deterministic One-state MDP

If $r(a_i)$ is **deterministic** and **known**, then

Optimal policy: $\pi^*(\mathbf{x}_0) = \text{argmax}_{a_i \in \mathbf{A}} \, r(a_i)$

What if reward function r is unknown?

Deterministic One-state MDP

If $r(a_i)$ is **deterministic** and **unknown**, then

Algorithm:

1. for each $a_i \in \mathbf{A}$
   - **execute** a_i and **collect** reward $r_{(i)}$
2. Optimal policy: $\pi^*(\mathbf{x}_0) = a_i$, with $i = \text{argmax}_{i=1...|\mathbf{A}|} r_{(i)}$

Note: exactly $|\mathbf{A}|$ iterations are needed.

Non-Deterministic One-state MDP

If $r(a_i)$ is **non-deterministic** and **known**, then

Optimal policy: $\pi^*(\mathbf{x}_0) = \text{argmax}_{a_i \in \mathbf{A}} E[r(a_i)]$

Example:

If $r(a_i) = \mathcal{N}(\mu_i, \sigma_i)$, then

$\pi^*(\mathbf{x}_0) = a_i$, with $i = \text{argmax}_{i=1...|\mathbf{A}|} \mu_i$

Non-Deterministic One-state MDP

If $r(a_i)$ is **non-deterministic** and **unknown**, then

Algorithm:

1. Initialize a data structure Θ
2. For each time $t = 1, \ldots, T$ (until termination condition)
   - **choose** an action $a_{(t)} \in \mathbf{A}$
   - **execute** $a_{(t)}$ and **collect** reward $r_{(t)}$
   - Update the data structure Θ
3. Optimal policy: $\pi^*(\mathbf{x}_0) = \ldots$, according to the data structure Θ

Note: **many** iterations ($T >> |\mathbf{A}|$) are needed.

Non-Deterministic One-state MDP

Example:

If $r(a_i)$ is **non-deterministic** and **unknown** and $r(a_i) = \mathcal{N}(\mu_i, \sigma_i)$, then

Algorithm:

1. Initialize $\Theta_{(0)}[i] \leftarrow 0$ and $c[i] \leftarrow 0$, $i = 1...|\mathbf{A}|$
2. For each time $t = 1, \ldots, T$ (until termination condition)
   - **choose** an index $\hat{\imath}$ for action $a_{(t)} = a_{\hat{\imath}} \in \mathbf{A}$
   - **execute** $a_{(t)}$ and **collect** reward $r_{(t)}$
   - increment $c[\hat{\imath}]$
   - update $\Theta_{(t)}[\hat{\imath}] \leftarrow \frac{1}{c[\hat{\imath}]} \left(r_{(t)} + (c[\hat{\imath}] - 1)\Theta_{(t-1)}[\hat{\imath}] \right)$
3. Optimal policy: $\pi^*(\mathbf{x}_0) = a_i$, with $i = \operatorname{argmax}_{i=1...|\mathbf{A}|} \Theta_{(T)}[i]$

Learning with Markov Decision Processes

Given an agent accomplishing a task according to an MDP $\langle \mathbf{X}, \mathbf{A}, \delta, r \rangle$, for which functions δ and r are **unknown** to the agent,

determine the optimal policy π^*

Note: This is not a supervised learning approach!

- Target function is $\pi : \mathbf{X} \rightarrow \mathbf{A}$
- but we do not have training examples $\{(\mathbf{x}_{(i)}, \pi(\mathbf{x}_{(i)}))\}$
- training examples are in the form $\langle (\mathbf{x}_{(1)}, a_{(1)}, r_{(1)}), \ldots, (\mathbf{x}_{(t)}, a_{(t)}, r_{(t)}) \rangle$

Agent's Learning Task

Since δ and r are not known, the agent cannot predict the effect of its actions. But it can execute them and then observe the outcome.

The learning task is thus performed by repeating these steps:

- **choose** an action
- **execute** the chosen action
- **observe** the resulting new state
- **collect** the reward

Approaches to Learning with MDP

- Value iteration
  (estimate the Value function and then compute π)
- Policy iteration
  (estimate directly π)

Learning through value iteration

The agent could learn the value function $V^{\pi^*}(\mathbf{x})$ (written as $V^*(\mathbf{x})$)

From which it could determine the optimal policy:

$$\pi^*(\mathbf{x}) = \underset{a \in \mathbf{A}}{\operatorname{argmax}} \left[r(\mathbf{x}, a) + \gamma V^*(\delta(\mathbf{x}, a)) \right]$$

However, this policy cannot be computed in this way because δ and r are not known.

Q Function

$Q^\pi(\mathbf{x}, a)$: expected value when executing a in the state $\mathbf{x}$ and then act according to π.

$$Q^\pi(\mathbf{x}, a) \equiv r(\mathbf{x}, a) + \gamma V^\pi(\delta(\mathbf{x}, a))$$

$$Q^\pi(\mathbf{x}, a) \equiv \sum_{\mathbf{x}'} P(\mathbf{x}'|\mathbf{x}, a)(r(\mathbf{x}, a, \mathbf{x}') + \gamma V^\pi(\mathbf{x}'))$$

If the agent learns Q, then it can determine the optimal policy without knowing δ and r.

$$\pi^*(\mathbf{x}) = \underset{a \in \mathbf{A}}{\operatorname{argmax}} Q(\mathbf{x}, a)$$

Training Rule to Learn Q

Deterministic case:

$$Q(\mathbf{x}_t, a_t) = r(\mathbf{x}_t, a_t) + \gamma \max_{a' \in \mathbf{A}} Q(\mathbf{x}_{t+1}, a')$$

Let $\hat{Q}$ denote learner's current approximation to Q.

Training rule:

$$\hat{Q}(\mathbf{x}, a) \leftarrow \bar{r} + \gamma \max_{a'} \hat{Q}(\mathbf{x}', a')$$

where $\bar{r}$ is the immediate reward and $\mathbf{x}'$ is the state resulting from applying action a in state $\mathbf{x}$.

Q Learning Algorithm for Deterministic MDPs

1. for each $\mathbf{x}, a$ initialize table entry $\hat{Q}_{(0)}(\mathbf{x}, a) \leftarrow 0$
2. observe current state $\mathbf{x}$
3. for each time $t = 1, \ldots, T$ (until termination condition)
   - **choose** an action a
   - **execute** the action a
   - **observe** the new state $\mathbf{x}'$
   - **collect** the immediate reward $\bar{r}$
   - update the table entry for $\hat{Q}(\mathbf{x}, a)$ as follows:

$$\hat{Q}_{(t)}(\mathbf{x}, a) \leftarrow \bar{r} + \gamma \max_{a' \in \mathbf{A}} \hat{Q}_{(t-1)}(\mathbf{x}', a')$$

   - $\mathbf{x} \leftarrow \mathbf{x}'$
4. Optimal policy: $\pi^*(\mathbf{x}) = \text{argmax}_{a \in \mathbf{A}}\, Q_{(T)}(\mathbf{x}, a)$

Note: not using δ and r, but just observing new state $\mathbf{x}'$ and immediate reward $\bar{r}$, after the execution of the chosen action.

Experimentation Strategies

How actions are chosen by the agents?
Exploitation: select action a that maximizes $\hat{Q}(\mathbf{x}, a)$
Exploration: select action a with low value of $\hat{Q}(\mathbf{x}, a)$

ϵ-greedy strategy

Given, $0 \leq \epsilon \leq 1$,
select a random action with probability ϵ
select the best action with probability $1 - \epsilon$

ϵ can decrease over time (first exploration, then exploitation).

Experimentation Strategies

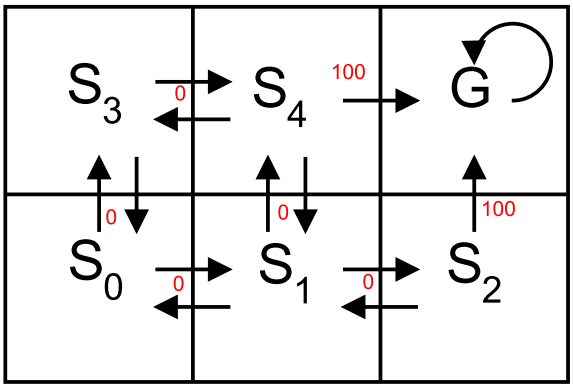
soft-max strategy

actions with higher $\hat{Q}$ values are assigned higher probabilities, but every action is assigned a non-zero probability.

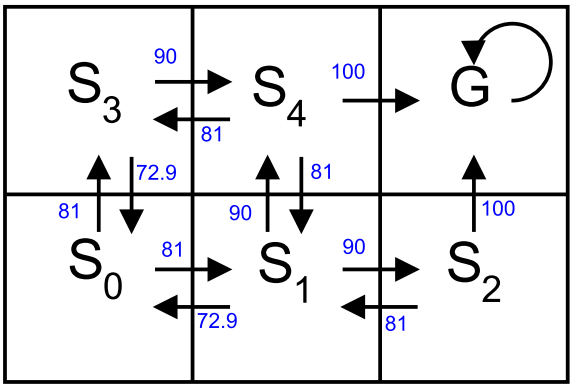
$$P(a_i|\mathbf{x}) = \frac{k^{\hat{Q}(\mathbf{x},a_i)}}{\sum_j k^{\hat{Q}(\mathbf{x},a_j)}}$$

$k > 0$ determines how strongly the selection favors actions with high $\hat{Q}$ values.
k may increase over time (first exploration, then exploitation).

Example: Grid World

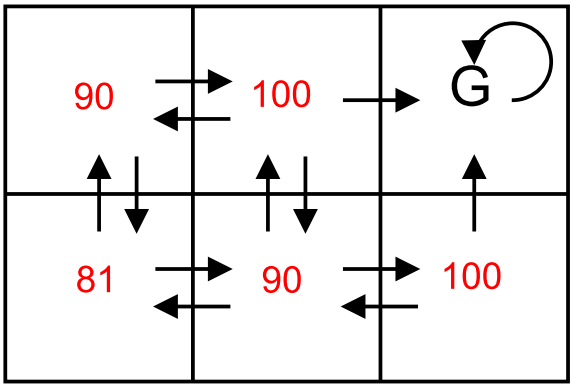


MDP

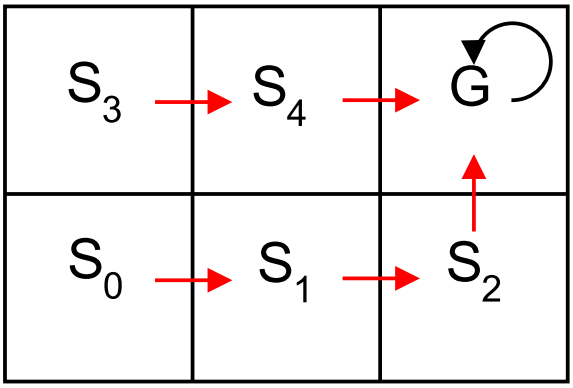


$\hat{Q}$

Example: Grid World

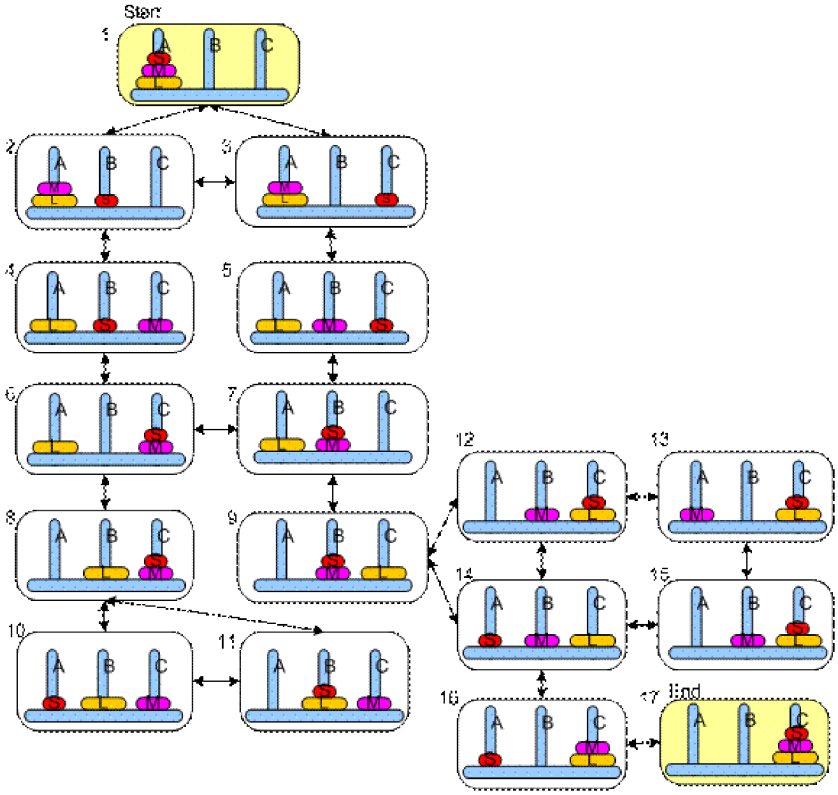


$V^*(\mathbf{x})$ values



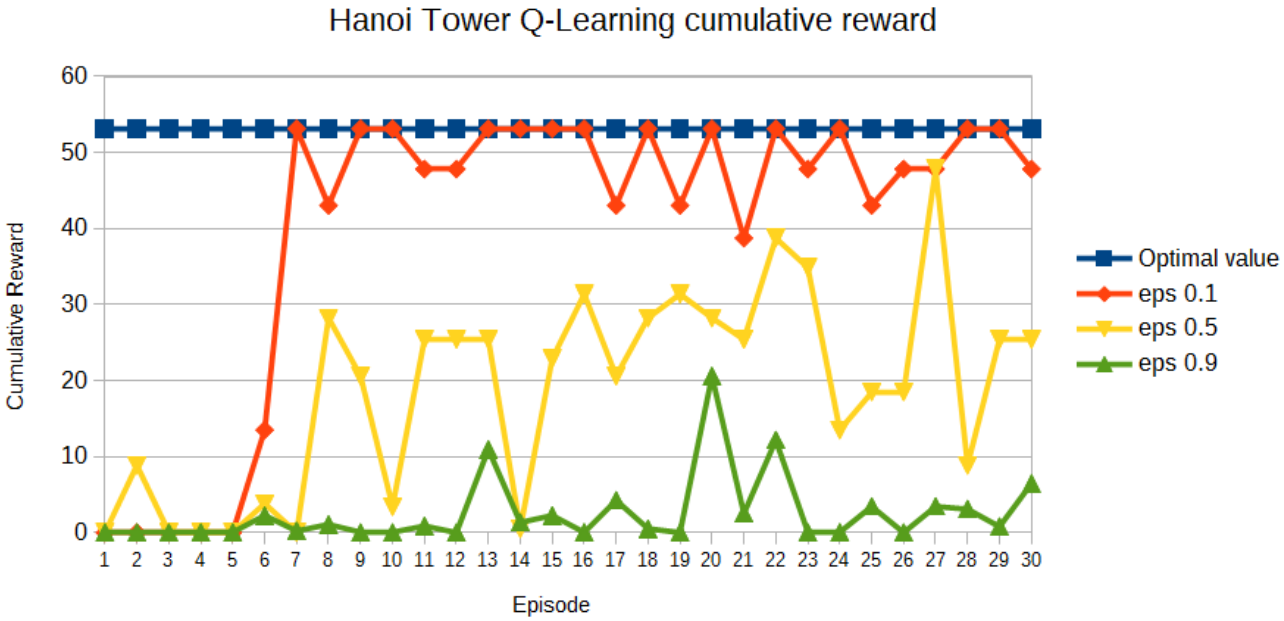
One optimal policy

Example: Hanoi Tower



Evaluating Reinforcement Learning Agents

Evaluation of RL agents is usually performed through the cumulative reward gained over time.



Evaluating Reinforcement Learning Agents

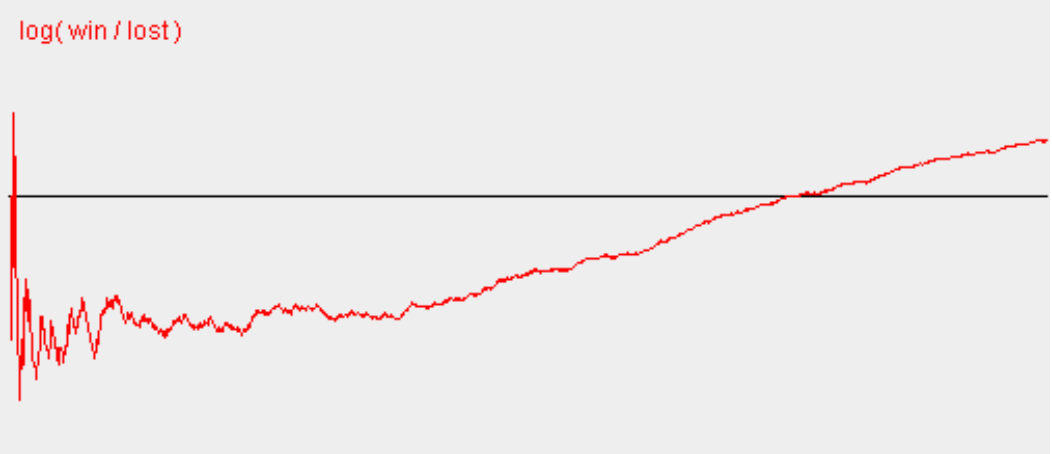
Cumulative reward plot may be very noisy (due to exploration phases).
A better approach could be:

Repeat until termination condition:

1. Execute k steps of learning
2. Evaluate the current policy π_k (average and stddev of cumulative reward obtained in d runs with no exploration)

Evaluating Reinforcement Learning Agents

Domain-specific performance metrics can also be used.



Average of all the results obtained during the learning process.