Brief introduction to deep reinforcement learning

Vincent François-Lavet

October 26th, 2018

Outline

Motivation for reinforcement learning

Machine learning relates to the capability of computers to **learn from examples** without following explicitly defined rules.

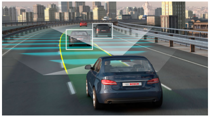
Three types of machine learning tasks can be described.

- Supervised learning is the task of inferring a classification or regression from labeled training data.
- Unsupervised learning is the task used to draw inferences from datasets consisting of input data without labeled responses.
- Reinforcement learning (RL) is the task concerned with how software agents ought to take actions in an environment in order to achieve some objectives.

Motivation

FIGURE – Example of an ATARI game : Seaquest

Motivation



Motivation

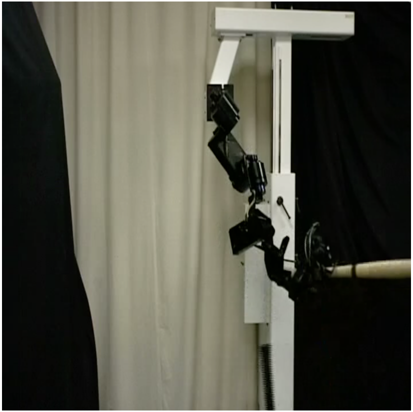
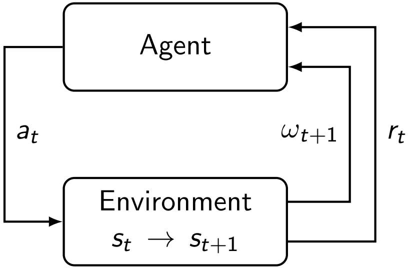


FIGURE – Application in robotics (credits : Jan Peters'team, Darmstadt)

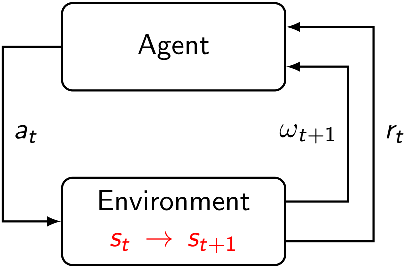
Objective

From experience in an environment,
an artificial agent
should be able to **learn** a sequential decision making task
in order **to achieve goals**.



Objective

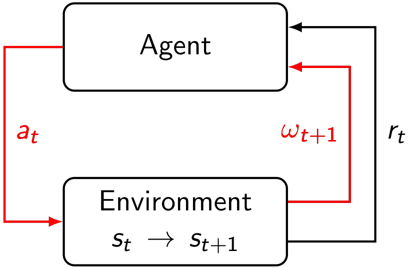
From experience in an environment,
an artificial agent
should be able to **learn** a sequential decision making task
in order **to achieve goals**.



transitions
are usually
stochastic

Objective

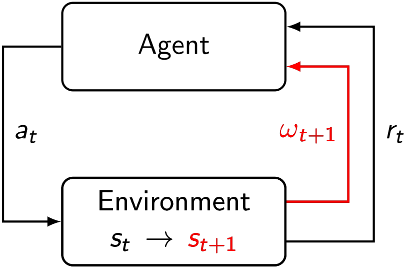
From experience in an environment,
an artificial agent
should be able to **learn** a sequential decision making task
in order **to achieve goals**.



Observations and
actions may be
high dimensional

Objective

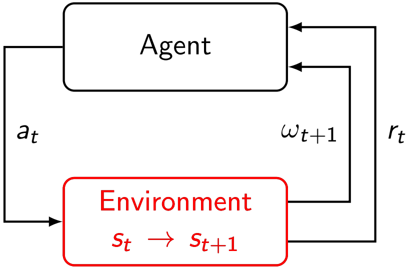
From experience in an environment,
an artificial agent
should be able to **learn** a sequential decision making task
in order **to achieve goals**.



Observations may not
provide full knowledge
of the underlying
state : $\omega_t \neq s_t$

Objective

From experience in an environment,
an artificial agent
should be able to **learn** a sequential decision making task
in order **to achieve goals**.



Experience may be constrained (e.g., not access to an accurate simulator or limited data)

Introduction

- Experience is gathered in the form of sequences of observations $\omega \in \Omega$, actions $a \in \mathcal{A}$ and rewards $r \in \mathbb{R}$:

$$\omega_0, a_0, r_0, ..., a_{t-1}, r_{t-1}, \omega_t$$

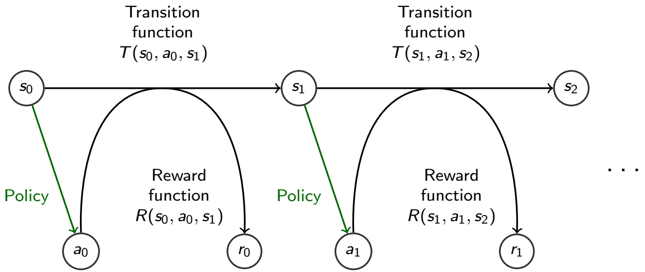
- In a fully observable environment, the state of the system $s_t \in \mathcal{S}$ is available to the agent.

$$s_t = \omega_t$$

Definition of an MDP

An MDP is a 5-tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$ where :

- $\mathcal{S}$ is a finite set of states $\{1, \dots, N_{\mathcal{S}}\}$,
- $\mathcal{A}$ is a finite set of actions $\{1, \dots, N_{\mathcal{A}}\}$,
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function (set of conditional transition probabilities between states),
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$ is the reward function, where $\mathcal{R}$ is a continuous set of possible rewards in a range $R_{max} \in \mathbb{R}^+$ (e.g., $[0, R_{max}]$),
- $\gamma \in [0, 1)$ is the discount factor.



Performance evaluation

In an MDP $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$, the expected return $V^{\pi}(s) : \mathcal{S} \to \mathbb{R}$ ($\pi \in \Pi$, e.g., $\mathcal{S} \to \mathcal{A}$) is defined such that

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, \pi\right], \tag{1}$$

with $\gamma \in [0, 1)$.

From the definition of the expected return, the optimal expected return can be defined as

$$V^*(s) = \max_{\pi \in \Pi} V^{\pi}(s). \tag{2}$$

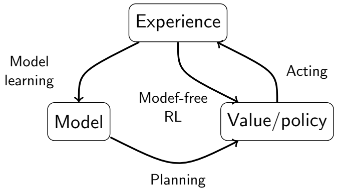
and the optimal policy can be defined as :

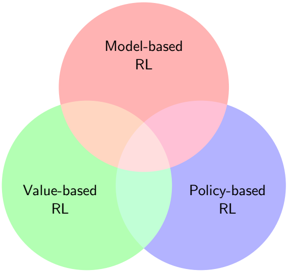
$$\pi^*(s) = \operatorname*{argmax}_{\pi \in \Pi} V^{\pi}(s). \tag{3}$$

Overview of deep RL

In general, an RL agent may include one or more of the following components :

- a representation of a value function that provides a prediction of how good is each state or each couple state/action,
- a direct representation of the policy $\pi(s)$ or $\pi(s, a)$, or
- a model of the environment in conjunction with a planning algorithm.





Deep learning has brought its generalization capabilities to RL.

Techniques used in deep reinforcement learning

Value-based methods

Value based methods : Q-learning

In addition to the V-value function, the Q-value function $Q^{\pi}(s, a) : \mathcal{S} \times A \to \mathbb{R}$ is defined as follows :

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum\nolimits_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a, \pi\right]. \qquad (4)$$

The particularity of the Q-value function as compared to the V-value function is that the optimal policy can be obtained directly from $Q^{*}(s, a)$:

$$\pi^{*}(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q^{*}(s, a). \qquad (5)$$

Value-based method : Q-learning with one entry for every state-action pair

In order to learn the optimal Q-value function, the Q-learning algorithm makes use of the Bellman equation for the Q-value function whose unique solution is $Q^*(s, a)$:

$$Q^*(s, a) = (\mathcal{B}Q^*)(s, a), \tag{6}$$

where $\mathcal{B}$ is the **Bellman operator** mapping any function $K : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ into another function $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and is defined as follows :

$$(\mathcal{B}K)(s, a) = \sum_{s' \in S} T(s, a, s') \left(R(s, a, s') + \gamma \max_{a' \in \mathcal{A}} K(s', a') \right). \tag{7}$$

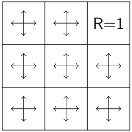
Value-based method : Q-learning (dynamic programming)

Value function $V = \max_a Q(s,a)$

Resulting policy $\pi = \text{argmax}_a Q(s,a)$

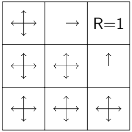
i=0

0	0	
0	0	0
0	0	0



i=1

0	0.9	
0	0	0.9
0	0	0



i=2

0.81	0.9	
0	0.81	0.9
0	0	0.81

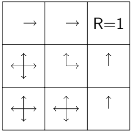


FIGURE – Grid-world MDP with $\gamma = 0.9$

Value-based method : Q-learning

In the tabular case :

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
    Initialize s
    Repeat (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

Q-learning with function approximator

To deal with continuous state and/or action space, we can represent value function with function approximators and parameters θ :

$$Q(s, a; \theta) \approx Q(s, a)$$

The parameters θ are updated such that :

$$\theta := \theta + \alpha \frac{d}{d\theta} \left(Q(s, a; \theta) - Y_k^Q \right)^2$$

with

$$Y_k^Q = r + \gamma \max_{a' \in \mathcal{A}} Q(s', a'; \theta_k).$$

With deep learning, the update usually uses a mini-batch (e.g., 32 elements) of tuples $< s, a, r, s' >$.

DQN algorithm

For Deep Q-Learning, we can represent value function by deep Q-network with weights θ (instabilities !). In the DQN algorithm :

- Replay memory
- Target network

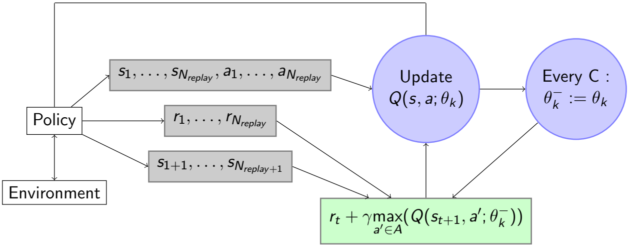


FIGURE – Sketch of the DQN algorithm. $Q(s, a; \theta_k)$ is initialized to random values (close to 0) everywhere on its domain and the replay memory is initially empty ; the target Q-network parameters θ_k^- are only updated every C iterations with the Q-network parameters θ_k and are held fixed between updates ; the update uses a mini-batch (e.g., 32 elements) of tuples $< s, a, r, s' >$ taken randomly in the replay memory.

Policy-based methods

Policy-based methods

- Parametrized policies $\Pi = \{\pi_w : w \in \mathbb{R}^n\}$.
- Policy search or gradient ascent on V^{π_w} to improve the policy.

- They are able to work with continuous action spaces. This is particularly interesting in applications such as robotics where forces and torques can take a continuum of values.
- They can represent stochastic policies : $\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{P}$. It is useful for building policies that can explicitly explore, and this is also useful in multi-agent systems (e.g., poker) where the Nash equilibrium is a stochastic policy.

Model-based methods

Model-based methods

The respective strengths of the model-free versus model-based approaches depend on different factors.

- If the agent does not have access to a generative model of the environment, the learned model will have some inaccuracies.
- Second, a model-based approach requires working in conjunction with a planning algorithm, which is often computationally demanding.
- Third, for some tasks, the model of the environment may be learned more efficiently due to the particular structure of the task.

Model-based methods

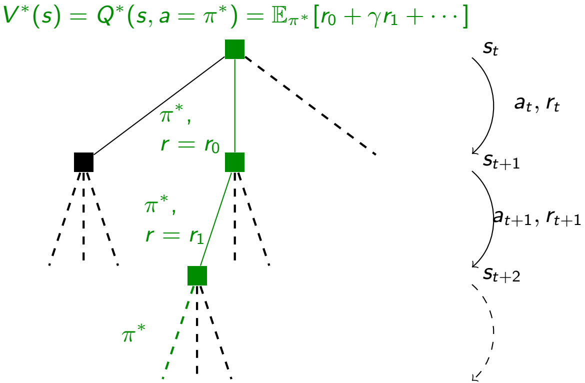
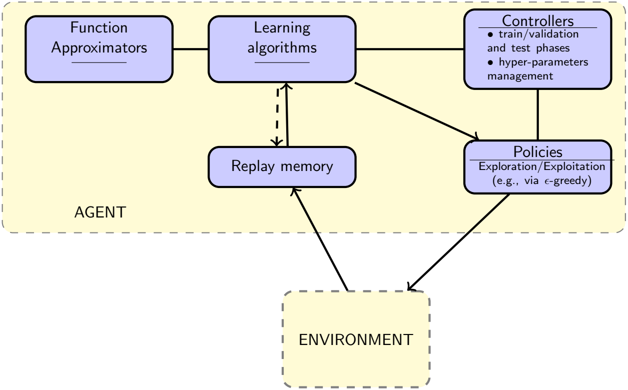


FIGURE – Illustration of model-based.

Overview



Implementation : https ://github.com/VinF/deer

Formalization

The learning algorithm can be seen as a mapping a dataset D_s into a policy π_{D_s} (independently of whether the policy comes from a model-based or a model-free approach) :

$$D_s \to \pi_{D_s}.$$

In an MDP, the suboptimality of the expected return can be decomposed as follows :

$$\underset{D_s \sim \mathcal{D}_s}{\mathbb{E}}[V^{\pi^*}(s) - V^{\pi_{D_s}}(s)] = \underbrace{(V^{\pi^*}(s) - V^{\pi_{D_s,\infty}}(s))}_{\text{asymptotic bias}} + \underbrace{\underset{D_s \sim \mathcal{D}_s}{\mathbb{E}}[(V^{\pi_{D_s,\infty}}(s) - V^{\pi_{D_s}}(s))}_{\substack{\text{error due to finite size of the dataset } D_s \\ \text{referred to as overfitting}}}]. \tag{8}$$

How to obtain the best policy ?

We can optimize the bias-overfitting tradeoff thanks to the following elements :

- the state representation,
- the objective function (e.g., reward shaping, tuning the training discount factor) and
- the learning algorithm (type of function approximator and model-free vs model-based).

And of course, if possible :

- improve the dataset (exploration/exploitation dilemma in an online setting)

Combining model-based and model-free via abstract representations

In cognitive science, there is a dichotomy between two modes of thoughts (*D. Kahneman. (2011). Thinking, Fast and Slow*) :

- a "System 1" that is fast and instinctive and
- a "System 2" that is slower and more logical.

FIGURE – System 1

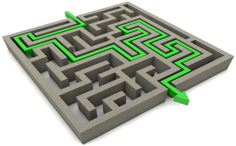


FIGURE – System 2

In deep reinforcement, a similar dichotomy can be observed when we consider the model-free and the model-based approaches.

Combining model-based and model-free

Learning everything through one abstract representation has the following advantages :

- it ensures that the features inferred in the abstract state provide good generalization ;
- it enables computationally efficient planning ;
- it facilitates interpretation of the decisions taken by the agent ;
- it allows developing new exploration strategies ;

Combined Reinforcement via Abstract Representations (CRAR)

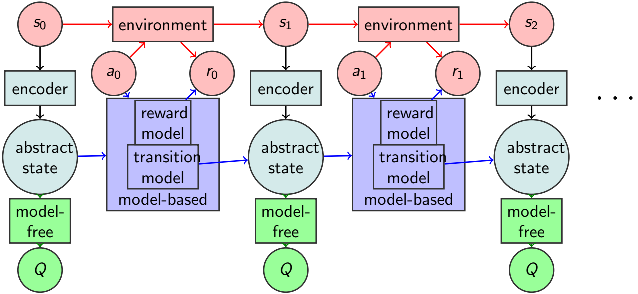


FIGURE – Illustration of the integration of model-based and model-free RL in the CRAR architecture, with a low-dimensional abstract state over which transitions and rewards are modeled.

The value function and the model are trained using off-policy data in the form of tuples (s, a, r, γ, s') via the abstract representation.

Another important challenge : transfer learning

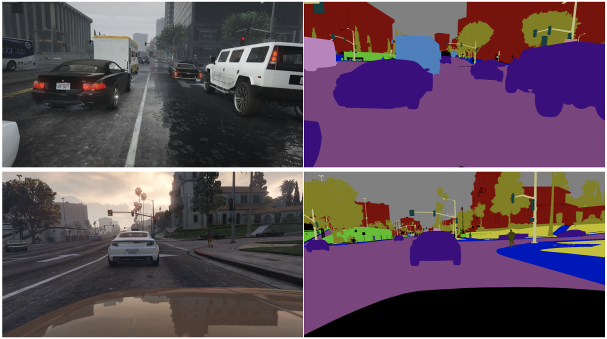


FIGURE – Transfer learning between different renderings. Picture from *"Playing for Data : Ground Truth from Computer Games", Richter, S. and Vineet, V., et al*

Transfer learning with the CRAR agent

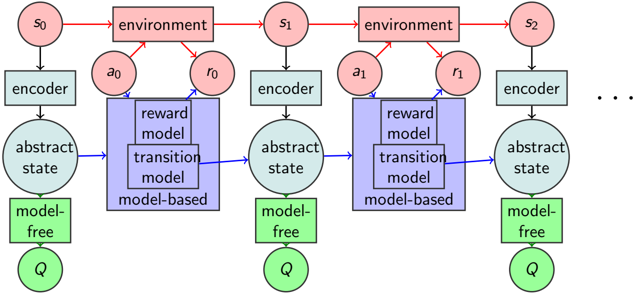


FIGURE – Illustration of the integration of model-based and model-free RL in the CRAR architecture, with a low-dimensional abstract state over which transitions and rewards are modeled.

Discussion of another parallel with neurosciences

How to discount deep RL

Motivations

Effect of the discount factor in an online setting.

- *Empirical studies of cognitive mechanisms in delay of gratification* : The capacity to wait longer for the preferred rewards seems to develop markedly only at about ages 3-4 ("marshmallow experiment").

Increasing discount factor (using the DQN aglorithm)

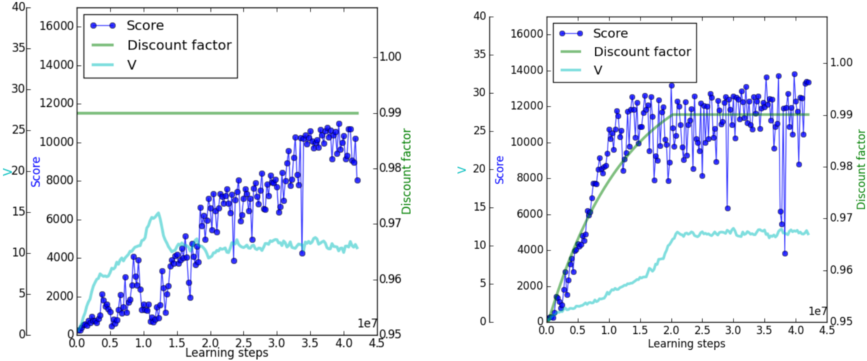


FIGURE – Illustration for the game q-bert of a discount factor γ held fixed on the right and an adaptive discount factor on the right.

Further ressources

- Sutton, Richard S., and Andrew G. Barto. Reinforcement learning : An introduction. Vol. 1. No. 1. Cambridge : MIT press, 1998.
- RL Course by David Silver on Youtube : https ://www.youtube.com/watch ?v=2pWv7GOvuf0

Conclusions

Summary of the talk

- Introduction to reinforcement learning and deep reinforcement learning
- Why combining model-free and model-based approaches
- Brief discussion on some relations to neuroscience

Questions ?