Brief introduction to reinforcement learning and deep reinforcement learning

Vincent François-Lavet

April 3rd and 8th, 2019

Who am I ?

I am currently a post-doctoral research fellow at McGill University/Mila, where I am working with Joelle Pineau and Doina Precup.

I'm interested in building artificial agents that can achieve complex tasks in high-dimensional environments. With that goal, I build deep reinforcement learning algorithms that can have a good generalization from limited experience.

Foreword

In these two lessons, we will introduce the **Reinforcement Learning (RL)** setting and we will cover elements of introduction to **deep reinforcement learning**, which is the combination of RL and deep learning.

The content of these two lessons and additional information are discussed in the following book :

V François-Lavet, et al.*"An introduction to deep reinforcement learning"*. 140 pages. Foundations and Trends in ML. `https://arxiv.org/abs/1811.12560`

Outline

Reinforcement learning within machine learning

From computer science to machine learning

- Around the 1940s, Alan Turing's theory of computation suggested that a machine (following a few properties of what is now called a Turing machine) could simulate any conceivable act of mathematical deduction when ignoring resource limitations.
- However, at the time it was not clear how a machine could learn any type of tasks from experience.

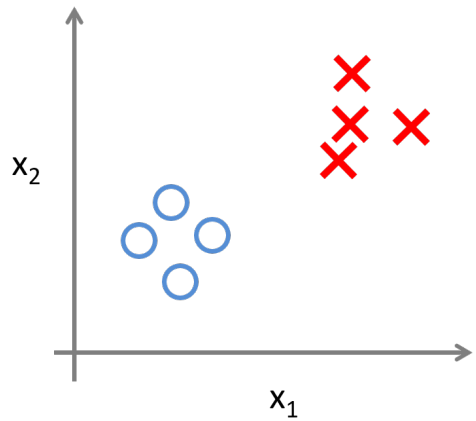
Machine learning is the area of computer science that relates to the capability of computers to **learn from examples**.

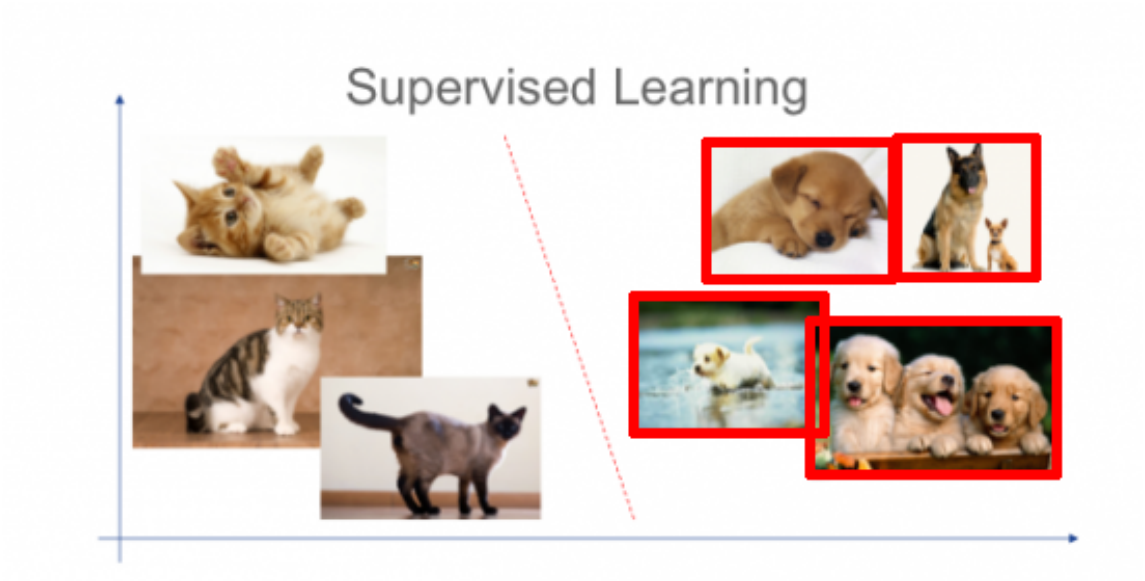
Three types of machine learning tasks can be described.

- **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 maximize cumulative (delayed) rewards.

Supervised learning

Supervised learning is the task of inferring a classification or regression from labeled training data.





Deep learning in supervised learning

In the field of supervised learning, deep learning has already exceeded human capabilities in most types of tasks where enough labelled data can be provided.

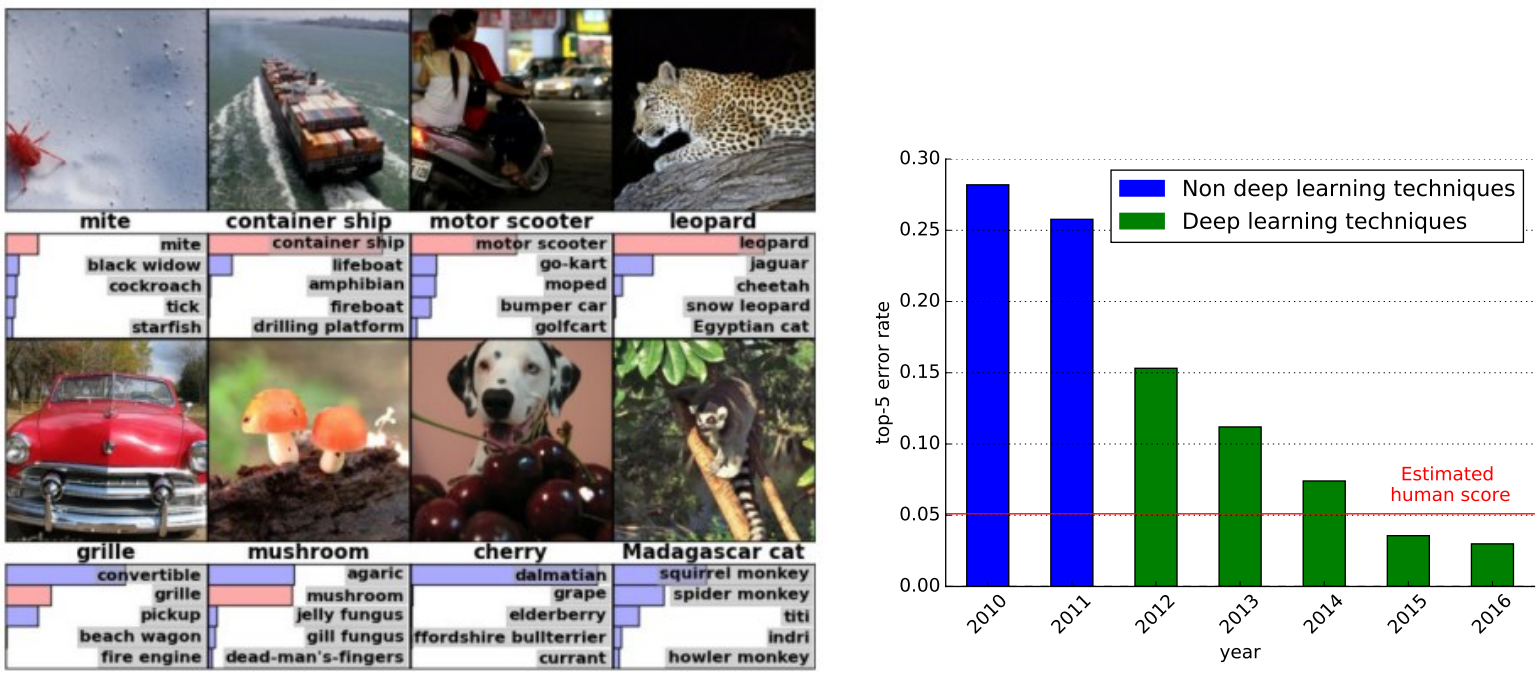
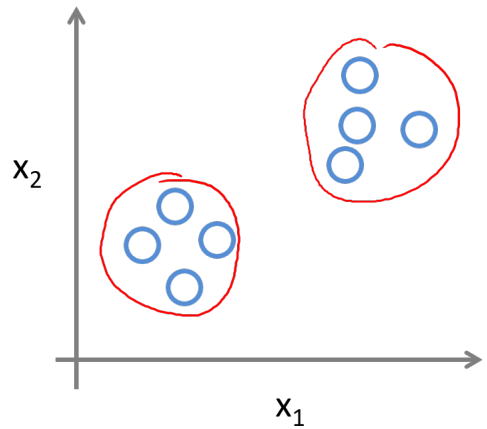


FIGURE – ILSVRC challenge.

Unsupervised learning

Unsupervised learning is the task used to draw inferences from datasets consisting of input data without labeled responses.



Deep learning in unsupervised learning

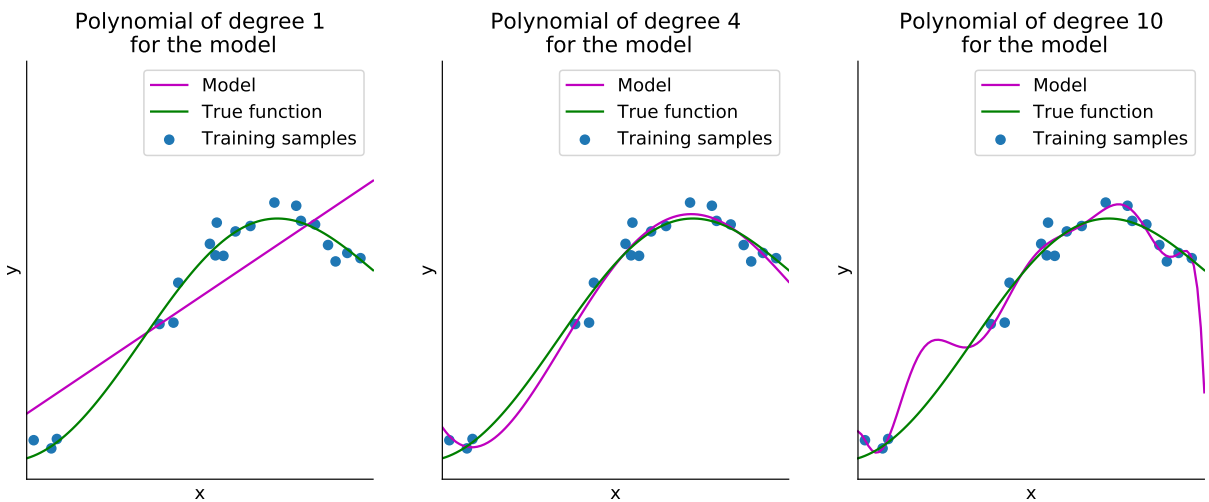
The latest results demonstrate that these types of algorithms achieve impressive results on image generation tasks with the Generative Adversarial Networks (GAN) architecture.

FIGURE – Sample of generated images in an unsupervised way from a dataset of 70K faces (Karras et al, 2018).

The deep learning representation

In its most abstract form, a neural network is a function $f : \mathcal{X} \to \mathcal{Y}$ parameterized with $\theta \in \mathbb{R}^{n_\theta}$ that takes as input $x \in \mathcal{X}$ and gives as output $y \in \mathcal{Y}$ ($\mathcal{X}$ and $\mathcal{Y}$ depend on the application) :

$$y = f(x; \theta). \tag{1}$$



Let us consider a simple neural network with one hidden fully-connected layer. The first layer is given the input values (i.e., the input features) x in the form of a column vector. The values of the next hidden layer are a transformation :

$$h = \text{ReLu}(W_1 \cdot x + b_1). \tag{2}$$

The hidden layer h can in turn be transformed. In this case :

$$y = (W_2 \cdot h + b_2). \tag{3}$$

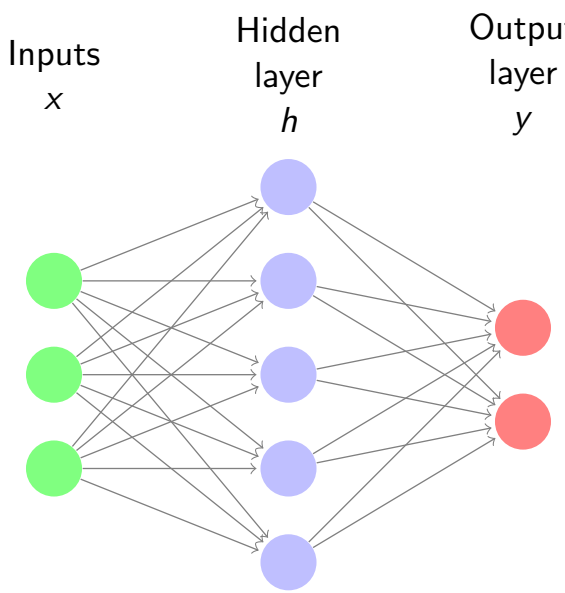


FIGURE – Example of a neural network with two fully-connected layers.

All the parameters of a neural network are trained in order to **minimize a cost function** (e.g., RMS error for regression or cross-entropy for classification).

The most common methods to learn the levels of abstraction in neural networks are based on **gradient descent** (the backpropagation algorithm), which allows the algorithm to change its internal parameters θ so as to fit the desired function.

The resurgence of interest in deep learning mainly comes from the following three aspects (which are complementary and lead to a virtuous circle) :

- an exponential increase of computational power (with the use of GPUs),
- methodological breakthroughs in deep learning and
- a growing eco-system of softwares and datasets.

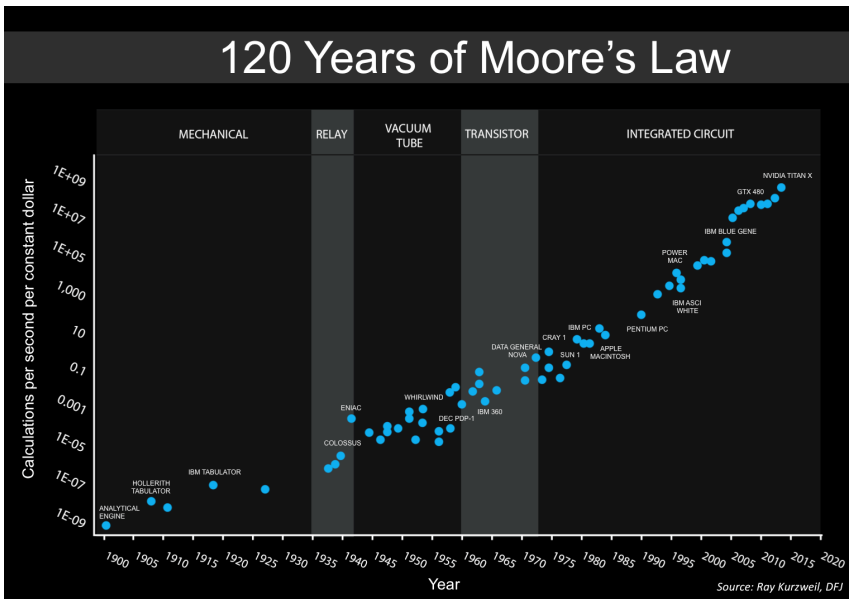


FIGURE – Illustration of the Moore's Law over 120 Years. The 7 most recent data points are all GPUs.

Motivation for reinforcement learning

Classic toy environment for Reinforcement Learning

A car tries to reach the top of the hill but the engine is not strong enough.

- State : position and velocity
- Action : accelerate forward, accelerate backward, coast.
- Goal : get the car to the top of the hill (e.g., reward $= 1$ at the top).

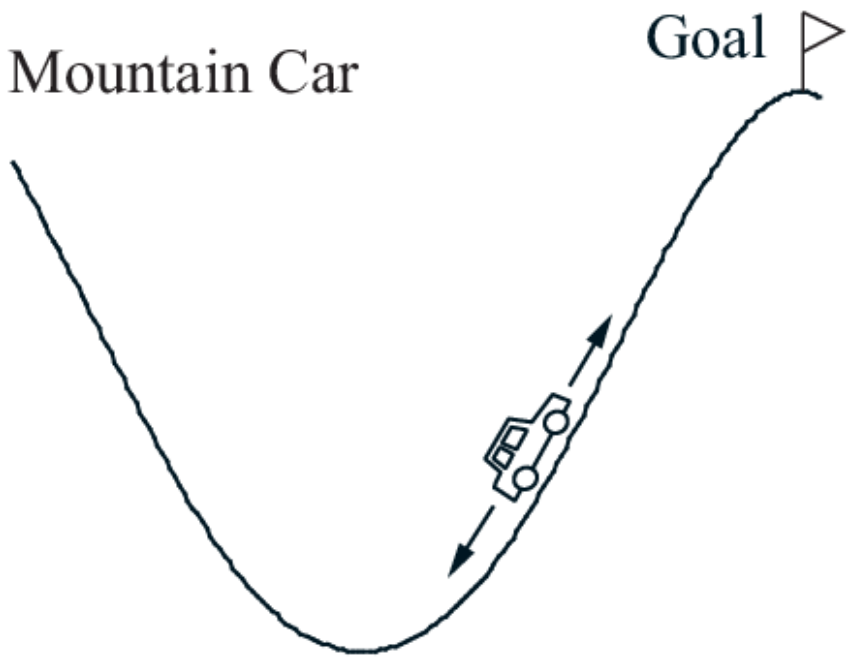


FIGURE – Mountain car

Classic toy environment for Reinforcement Learning

A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track.

- State : Position of the cart (x) and the pole (θ).
- Action : accelerate right, accelerate left, coast.
- Goal : keep the pole balanced (e.g., reward = -1 if it falls).

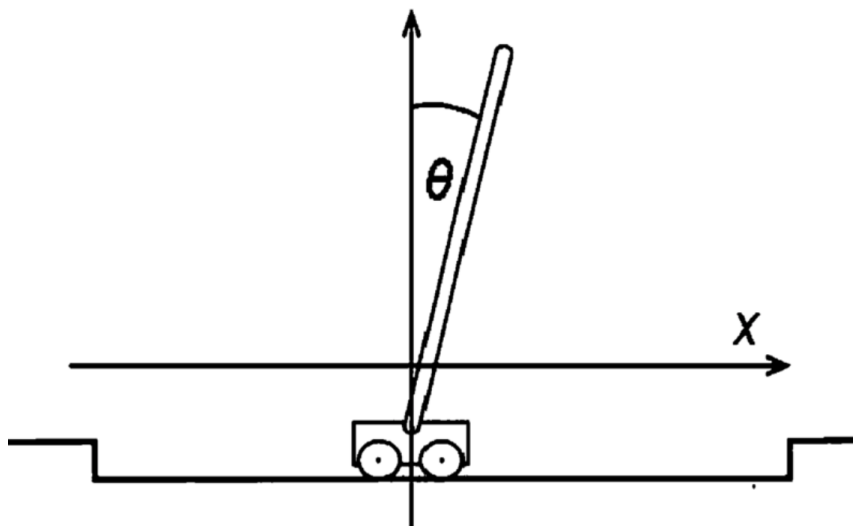
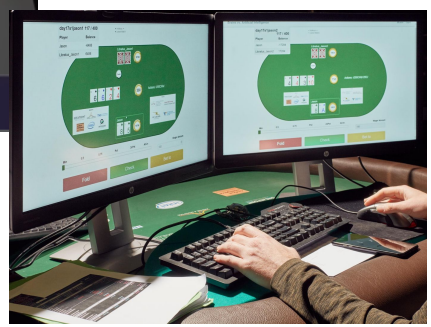


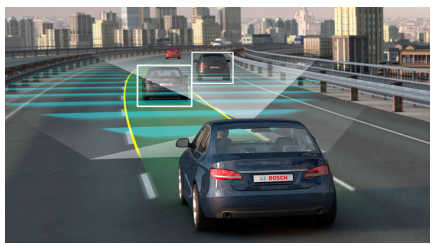
FIGURE – Cart pole

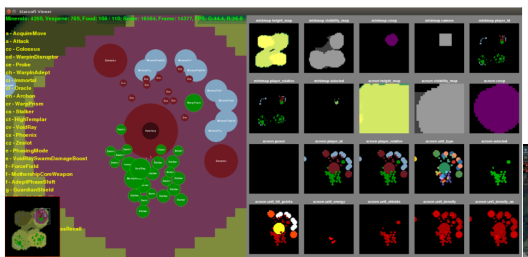
Motivation

FIGURE – Example of an ATARI game : Seaquest

Motivation : Overview







Motivation : Robotics

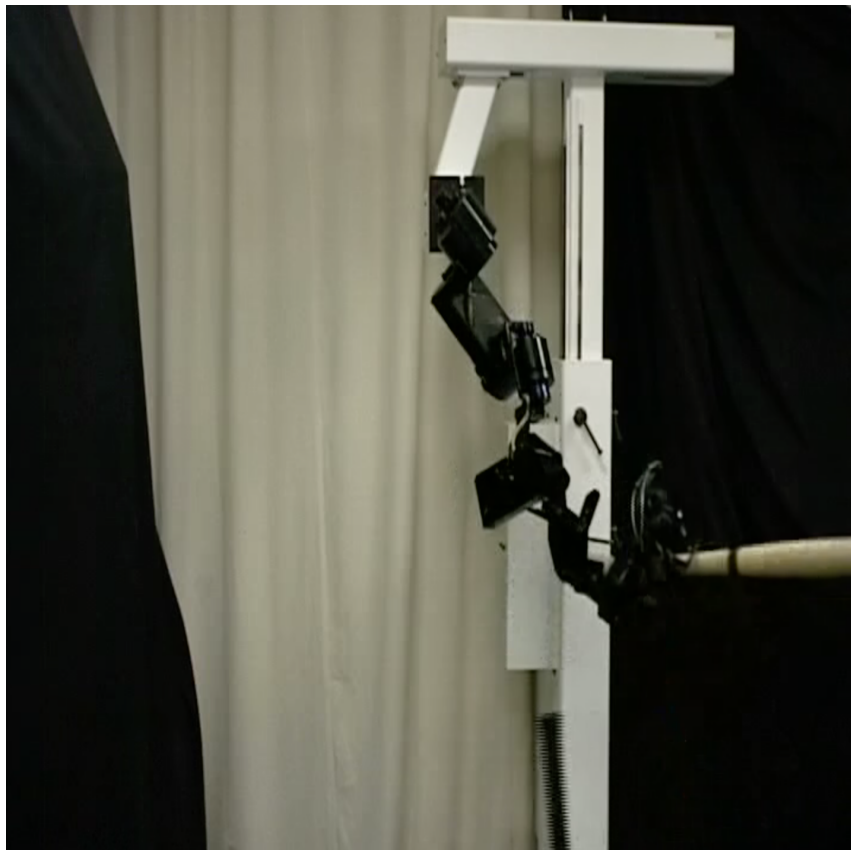
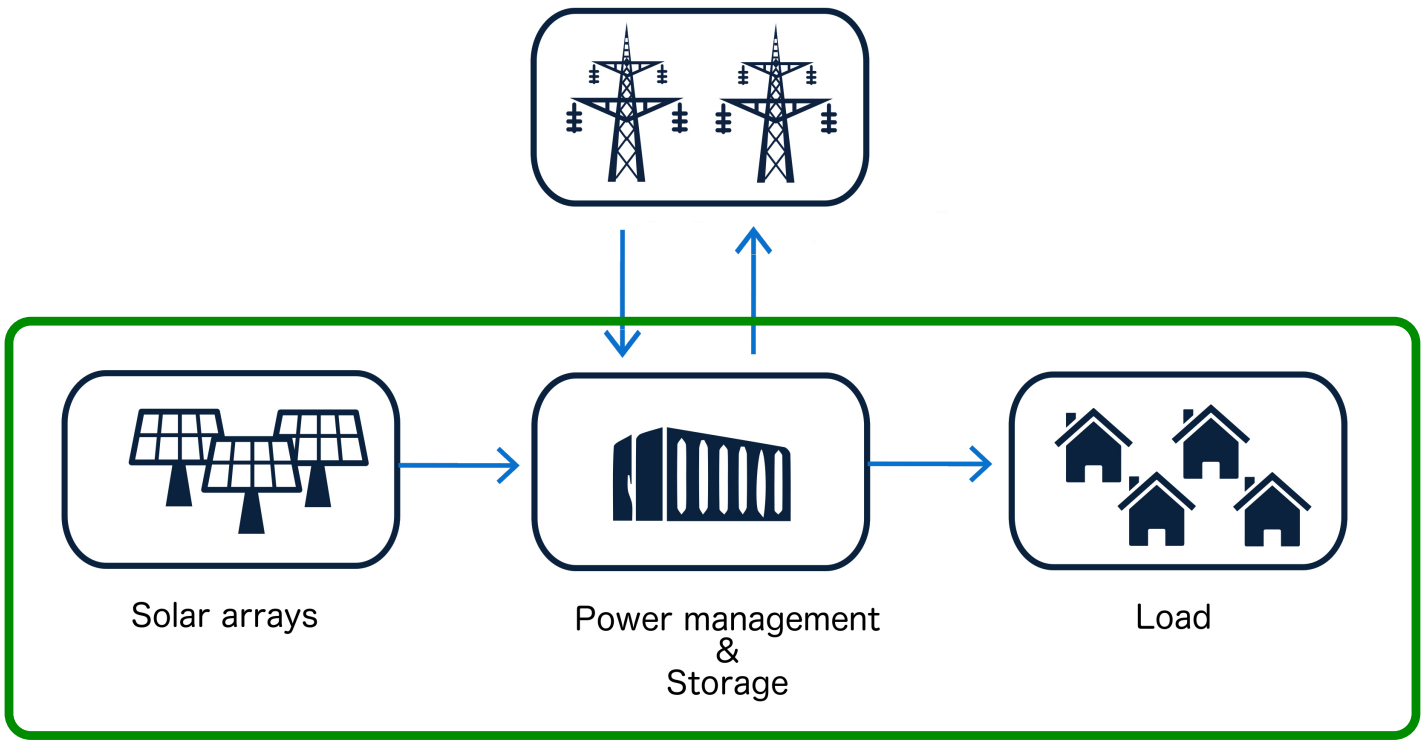


FIGURE – Application in robotics (credits : Jan Peters'team, Darmstadt)

Motivation : Smartgrids and microgrids

A microgrid is an electrical system that includes multiple loads and distributed energy resources that can be operated in parallel with the broader utility grid or as an electrical island.

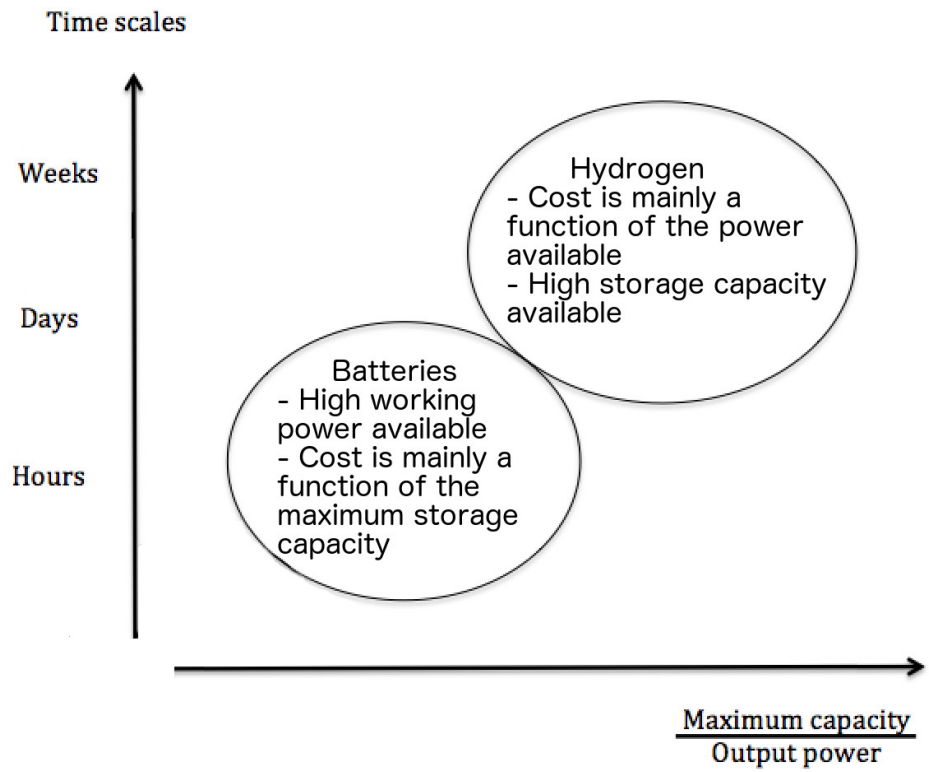


Microgrid

Microgrids and storage

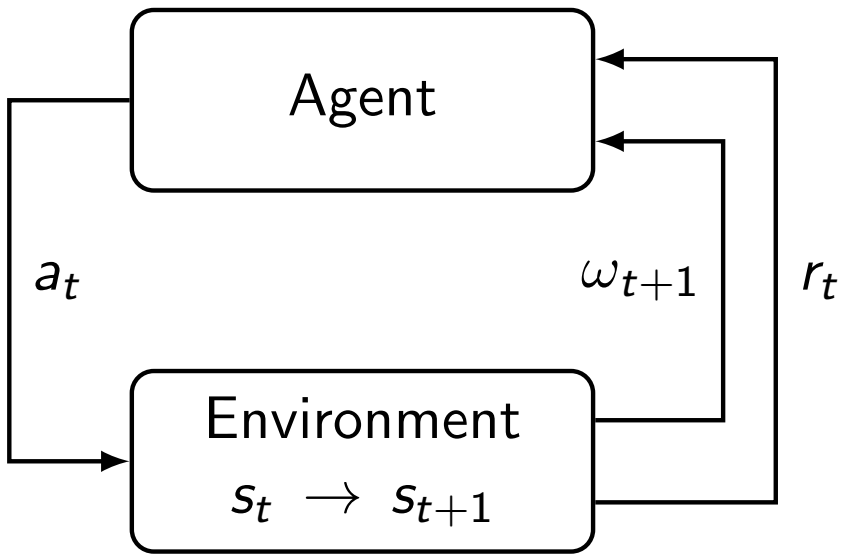
There exist opportunities with microgrids featuring :

- A short term storage capacity (typically batteries),
- A long term storage capacity (e.g., hydrogen).



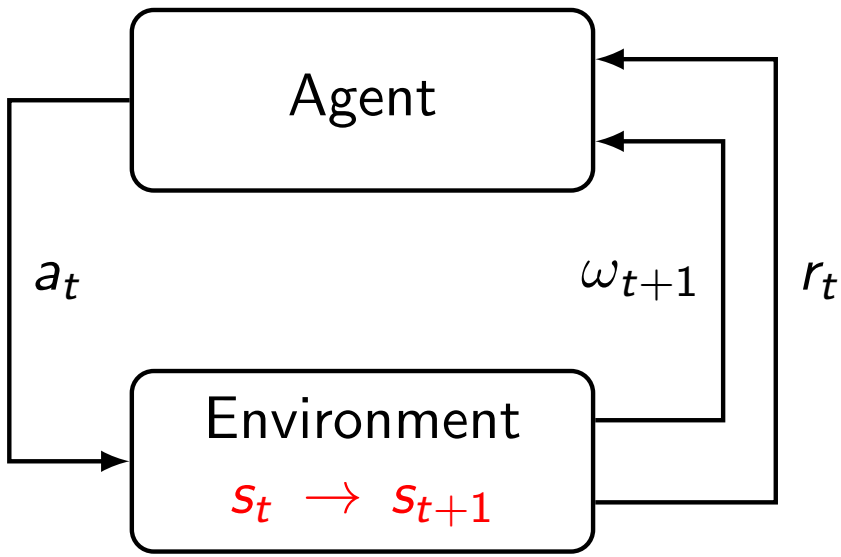
Objective

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



Objective

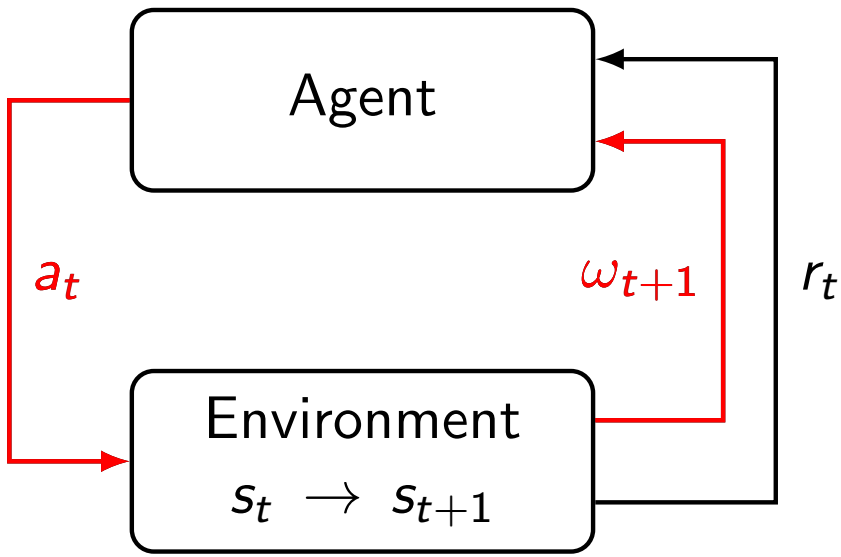
From experience in an environment,
an artificial agent
should be able to **learn** a sequential decision making task
in order **to achieve goals**.



transitions
are usually
stochastic

Objective

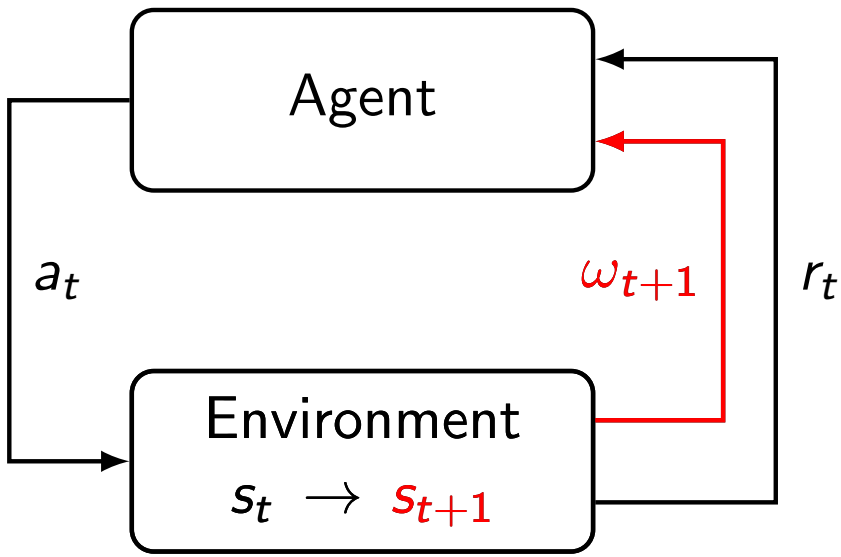
From experience in an environment,
an artificial agent
should be able to **learn** a sequential decision making task
in order **to achieve goals**.



Observations and
actions may be
high dimensional

Objective

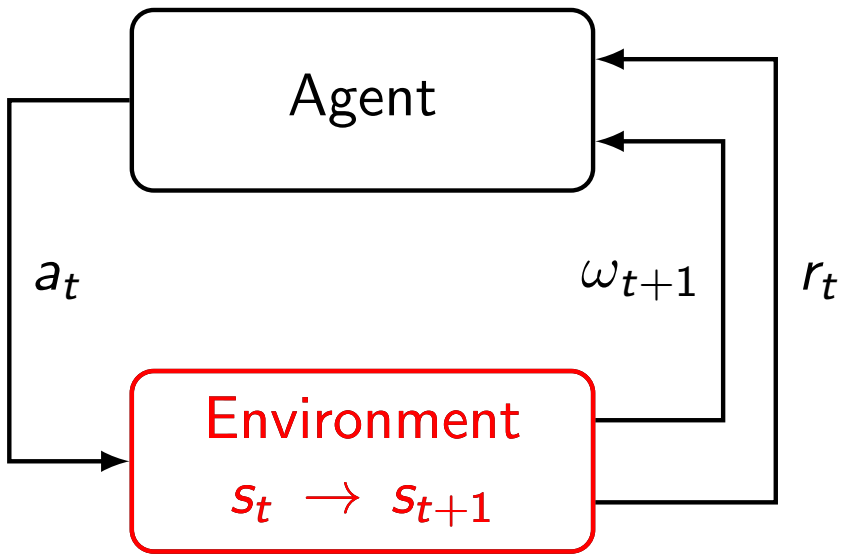
From experience in an environment,
an artificial agent
should be able to **learn** a sequential decision making task
in order **to achieve goals**.



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.

$$s_t = \omega_t$$

Example of a Markov Decision Process (MDP)

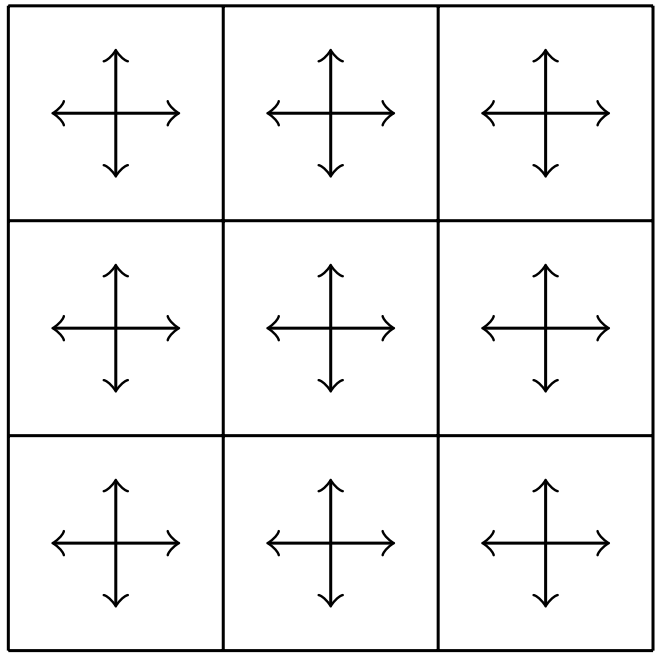
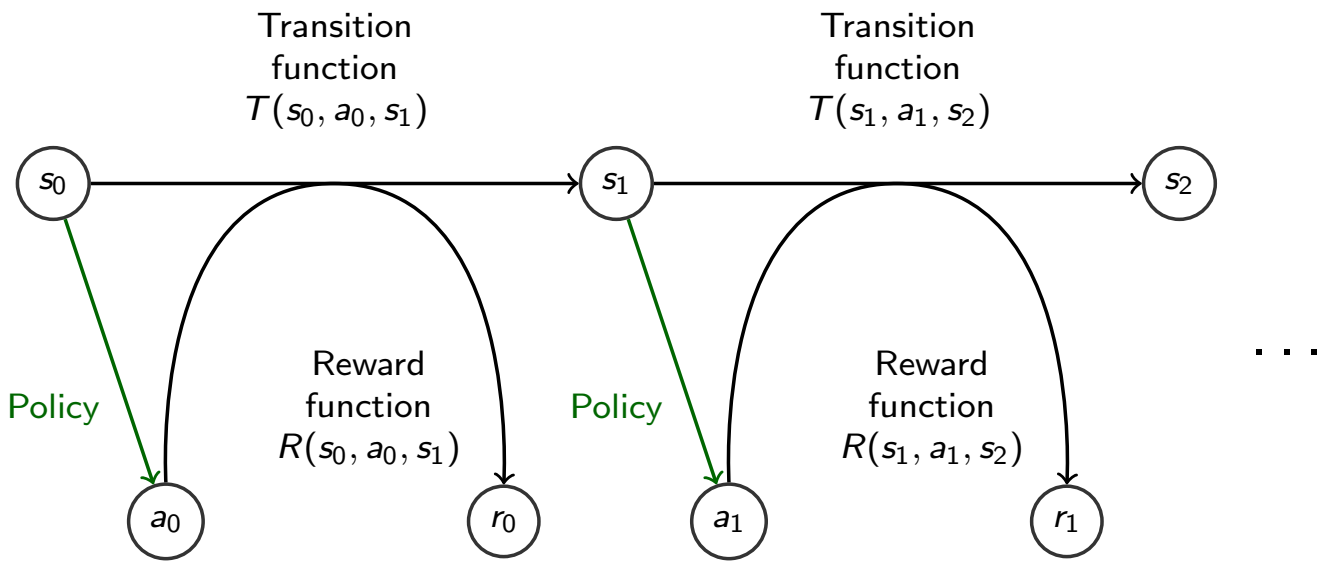


FIGURE – Representation of a mini grid-world with 9 discrete states and 4 discrete actions. The agent is able to move deterministically in the four directions, except when the agent is trying to get "out of the grid-world".

Definition of an MDP

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

- $\mathcal{S}$ is a finite set of states $\{1, \ldots, N_{\mathcal{S}}\}$,
- $\mathcal{A}$ is a finite set of actions $\{1, \ldots, N_{\mathcal{A}}\}$,
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ 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

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, \pi\right], \tag{4}$$

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

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

$$V^{*}(s) = \max_{\pi \in \Pi} V^{\pi}(s). \tag{5}$$

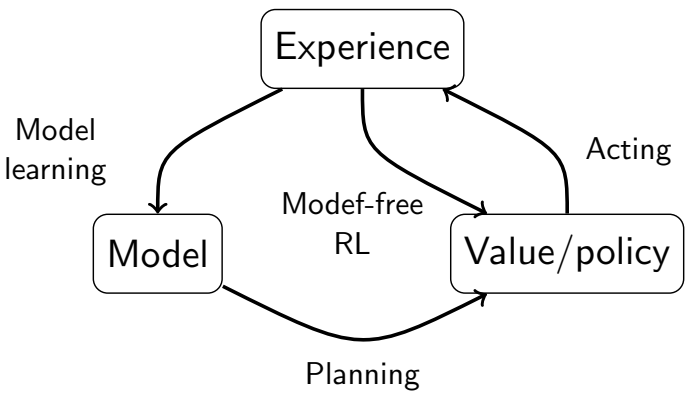
and the optimal policy can be defined as :

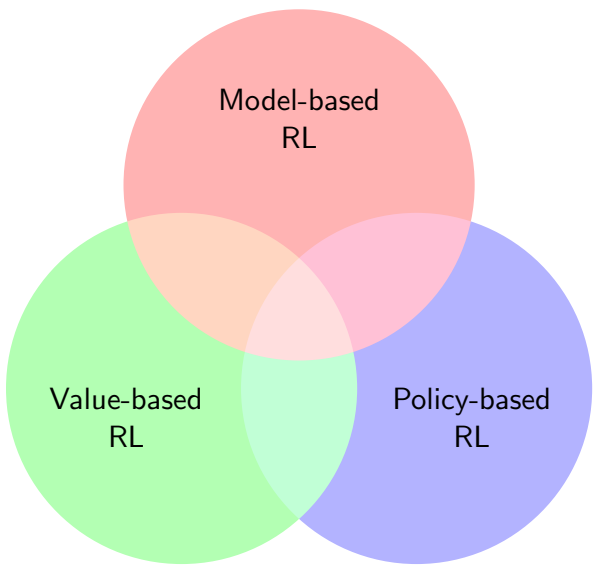
$$\pi^{*}(s) = \operatorname*{argmax}_{\pi \in \Pi} V^{\pi}(s). \tag{6}$$

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 $Q^{\pi}(s,a) : \mathcal{S} \times A \rightarrow \mathbb{R}$ is defined as follows :

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k} \mid s_t = s, a_t = a, \pi\right]. \tag{7}$$

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

$$\pi^{*}(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q^{*}(s,a). \tag{8}$$

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 $Q^*(s, a)$:

$$Q^*(s, a) = (\mathcal{B}Q^*)(s, a), \tag{9}$$

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

$$(\mathcal{B}K)(s, a) = \sum_{s' \in S} T(s, a, s') \left(R(s, a, s') + \gamma \max_{a' \in \mathcal{A}} K(s', a') \right). \tag{10}$$

Convergence of value iteration

Theorem : Value iteration with the Bellman operator converges to optimal values : $\hat{Q} \rightarrow Q^*$.

Sketch of the proof For any estimate of the value function $\hat{Q}$, we can show that the Bellman operator is a contraction, i.e. for any value function estimates $\hat{Q}_1$, $\hat{Q}_2$:

$$\|B\hat{Q}_1(s,a) - B\hat{Q}_2(s,a)\|_\infty \leq \gamma \|\hat{Q}_1(s,a) - \hat{Q}_2(s,a)\|_\infty$$

Since $BQ^* = Q^*$, the contraction property also implies convergence towards this unique fixed point.

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

At each iteration, let's do one step update from every state-action pair (s, a) :

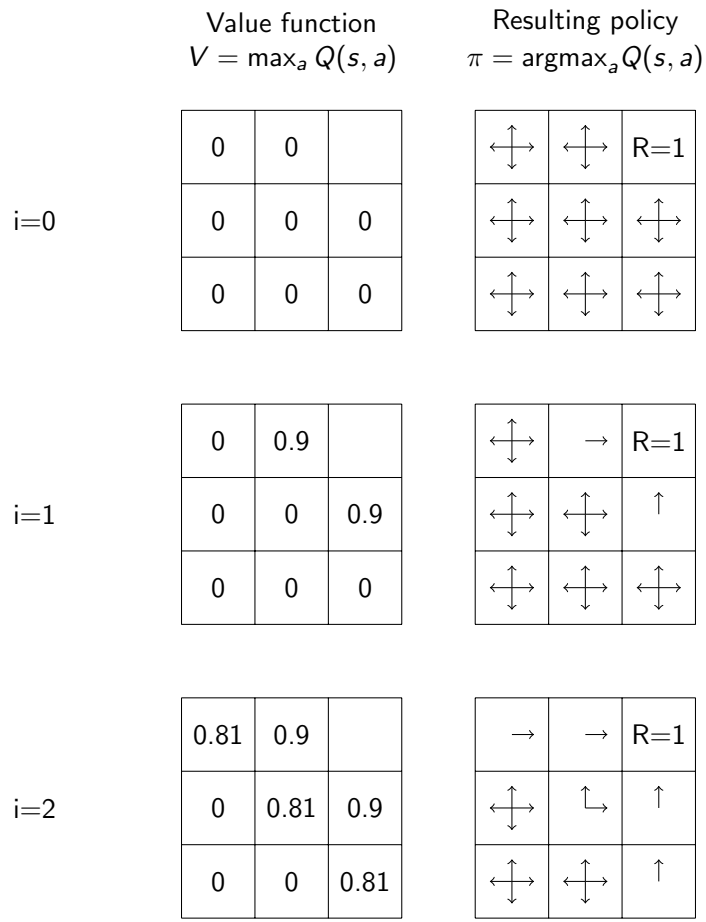


FIGURE – Grid-world MDP with $\gamma = 0.9$

Value-based method : Q-learning from tuples (s, a, r, s')

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'
    $Q(s, a) \leftarrow Q(s, a) + \alpha\left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$
    $s \leftarrow s'$;
  until s is terminal

Q-learning with function approximator

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

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

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

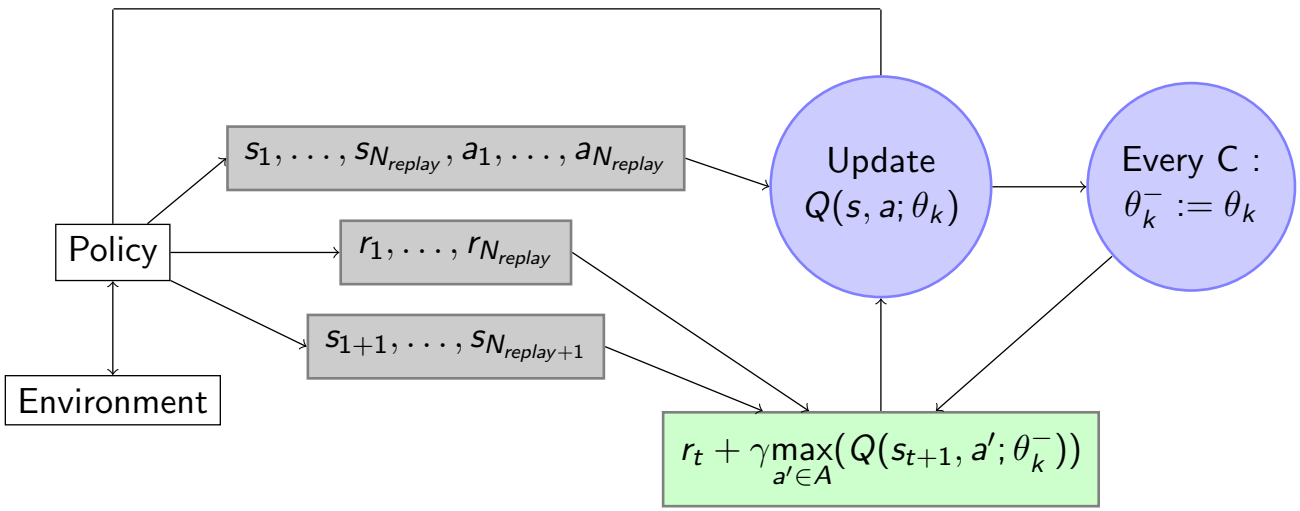
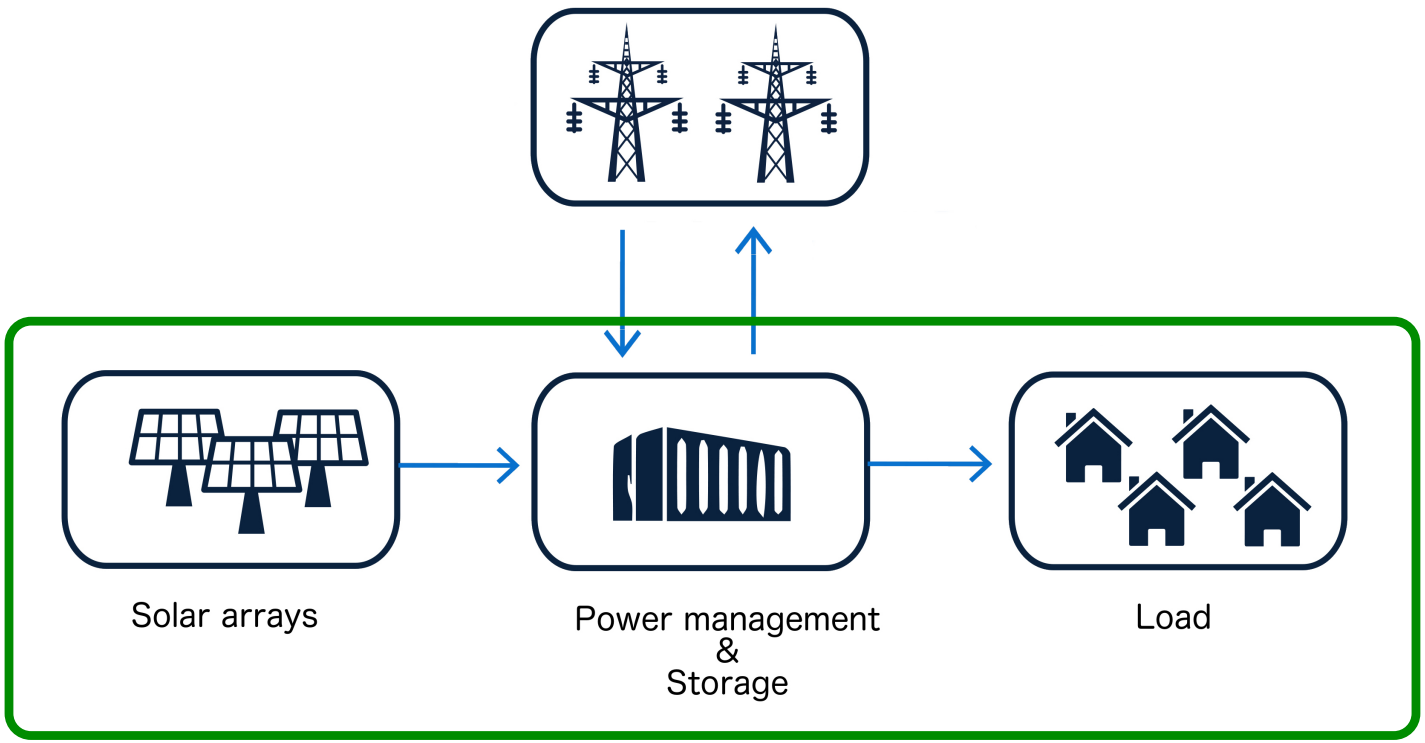


FIGURE – Sketch of the DQN algorithm. $Q(s, a; \theta_k)$ is initialized to random values (close to 0) everywhere on its domain and the replay memory is initially empty ; the target Q-network parameters θ_k^- are only updated every C iterations with the Q-network parameters θ_k and are held fixed between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

Application : optimizing the operation of a microgrid

A microgrid is an electrical system that includes multiple loads and distributed energy resources that can be operated in parallel with the broader utility grid or as an electrical island.



Microgrid

Structure of the Q-network

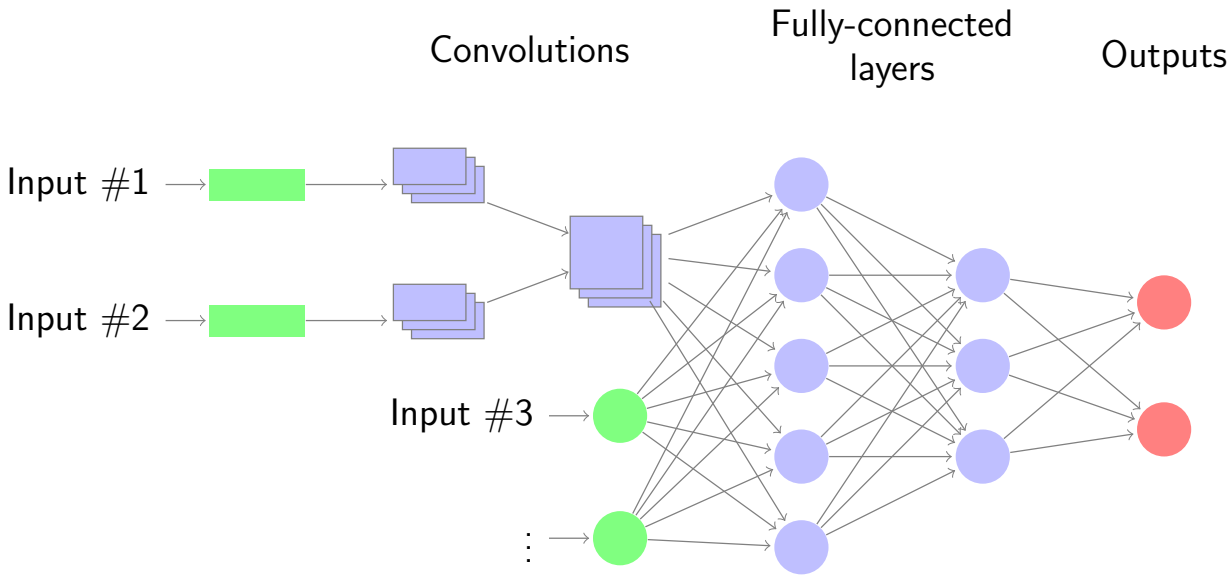
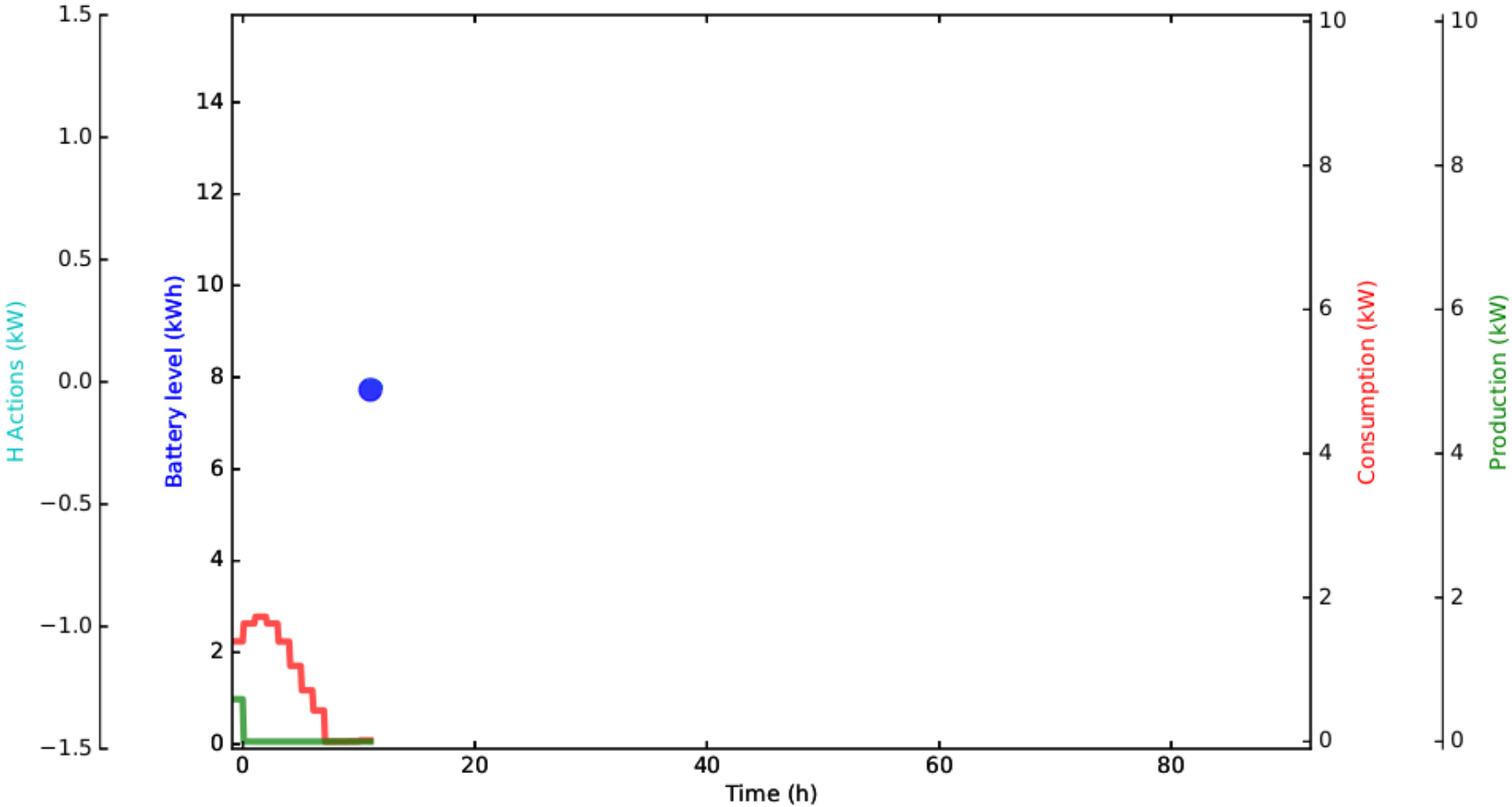


FIGURE – Sketch of the structure of the neural network architecture. The neural network processes the time series using a set of convolutional layers. The output of the convolutions and the other inputs are followed by fully-connected layers and the ouput layer. Architectures based on LSTMs instead of convolutions obtain similar results.

Example

Illustration of the policy on the test data with the following features :

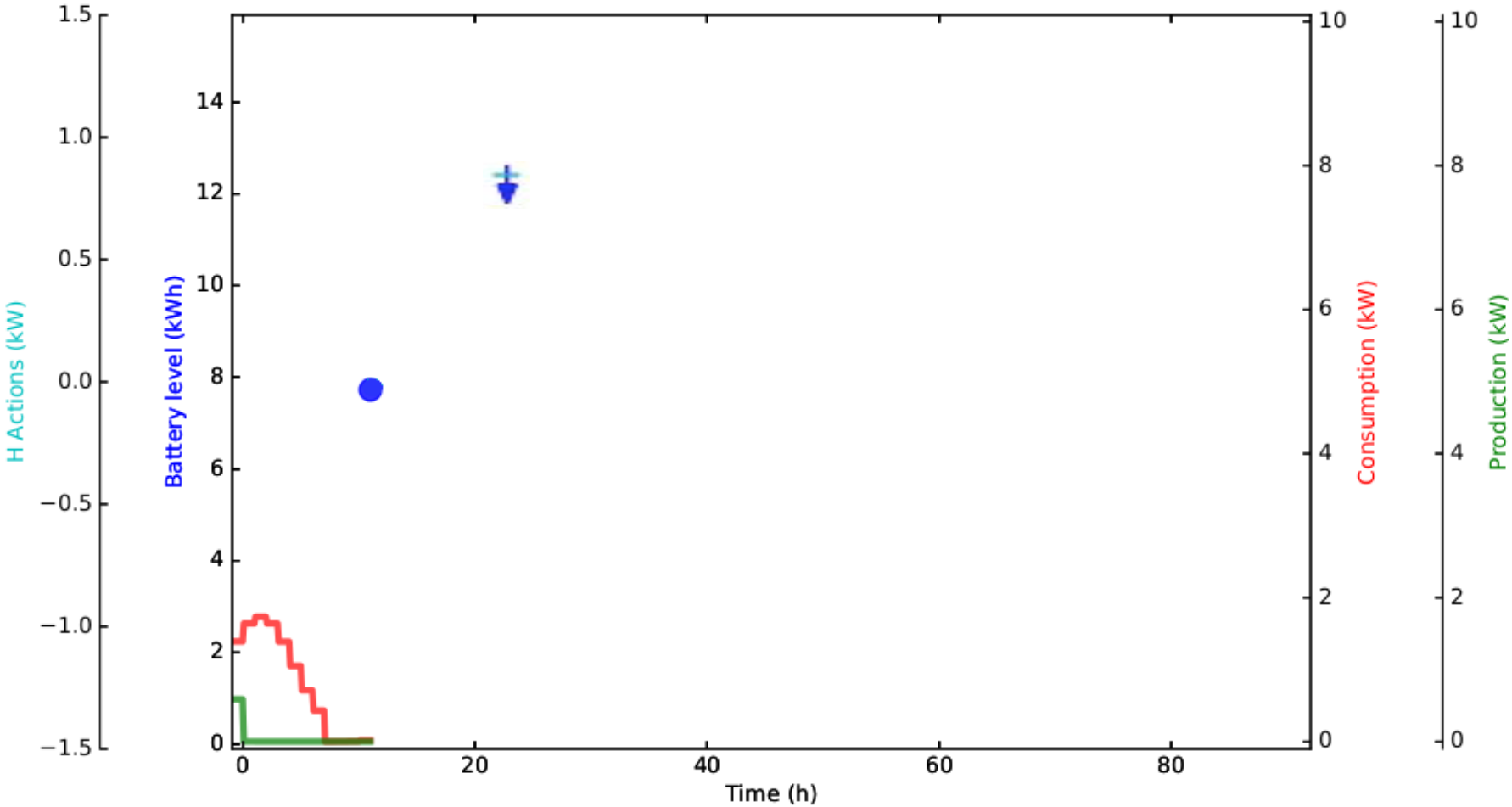
- past 12 hours for the production and consumption, and
- charge level of the battery.



Example

Illustration of the policy on the test data with the following features :

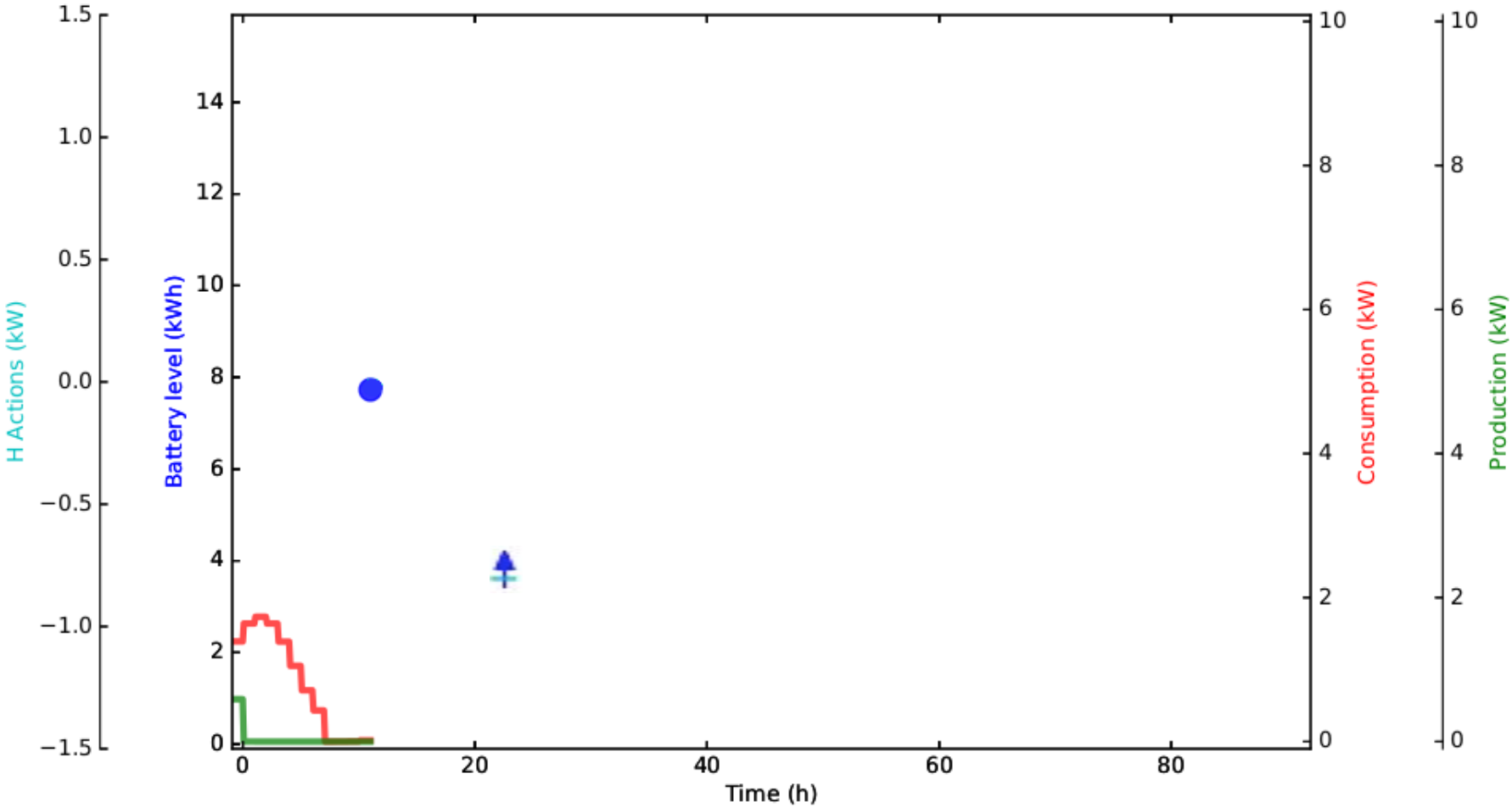
- past 12 hours for the production and consumption, and
- charge level of the battery.



Example

Illustration of the policy on the test data with the following features :

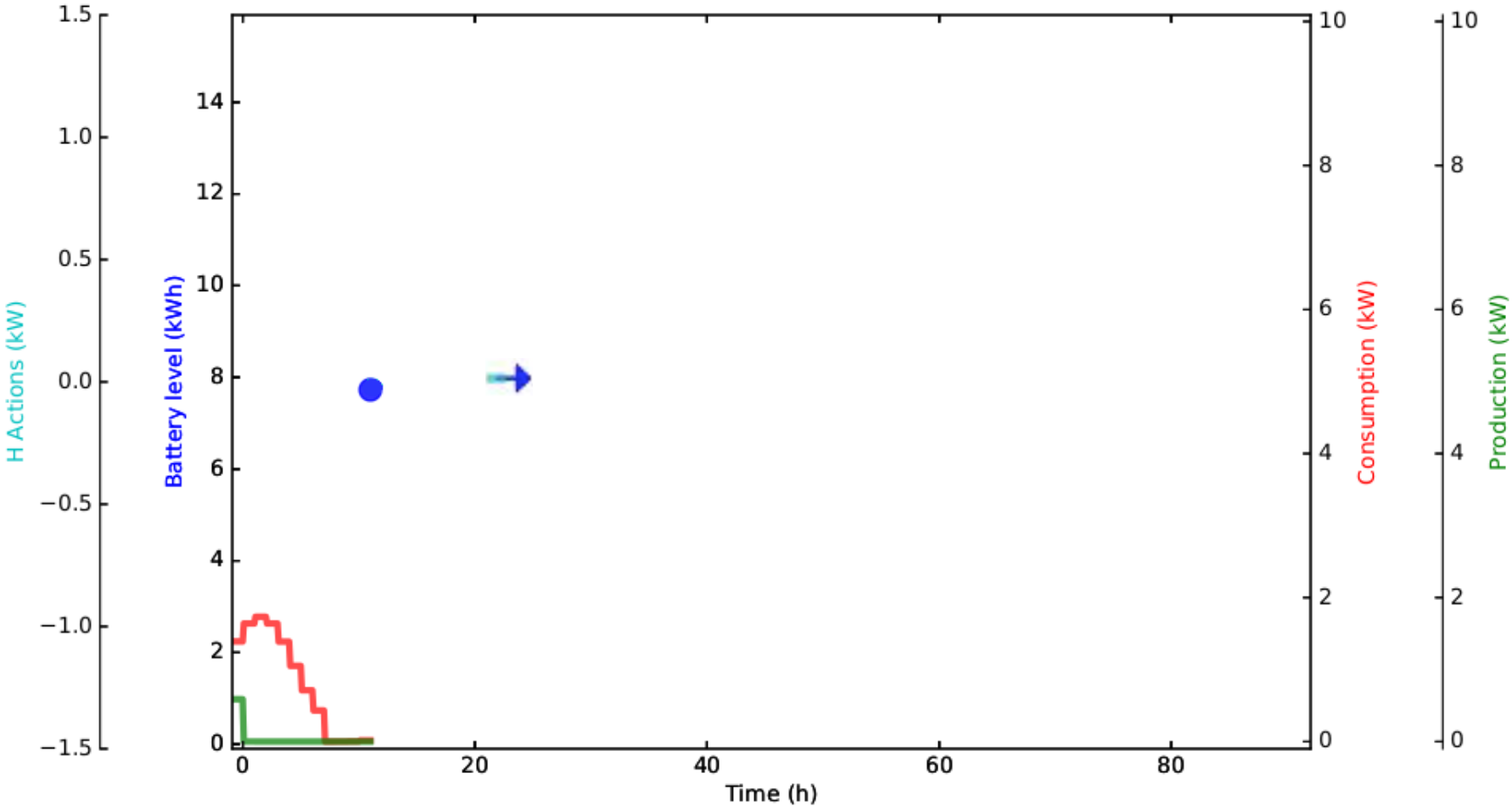
- past 12 hours for the production and consumption, and
- charge level of the battery.



Example

Illustration of the policy on the test data with the following features :

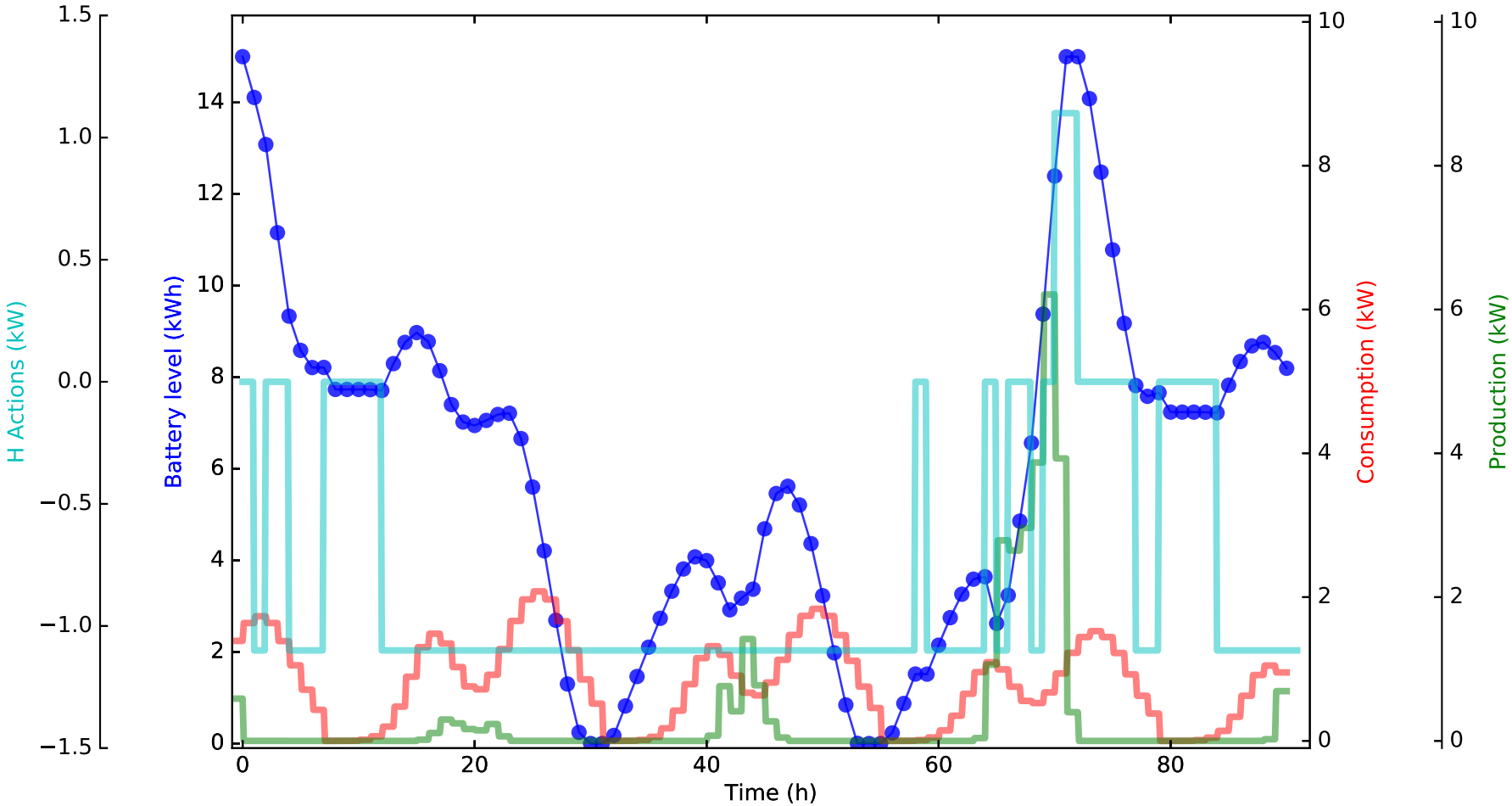
- past 12 hours for the production and consumption, and
- charge level of the battery.



Example

Illustration of the policy on the test data with the following features :

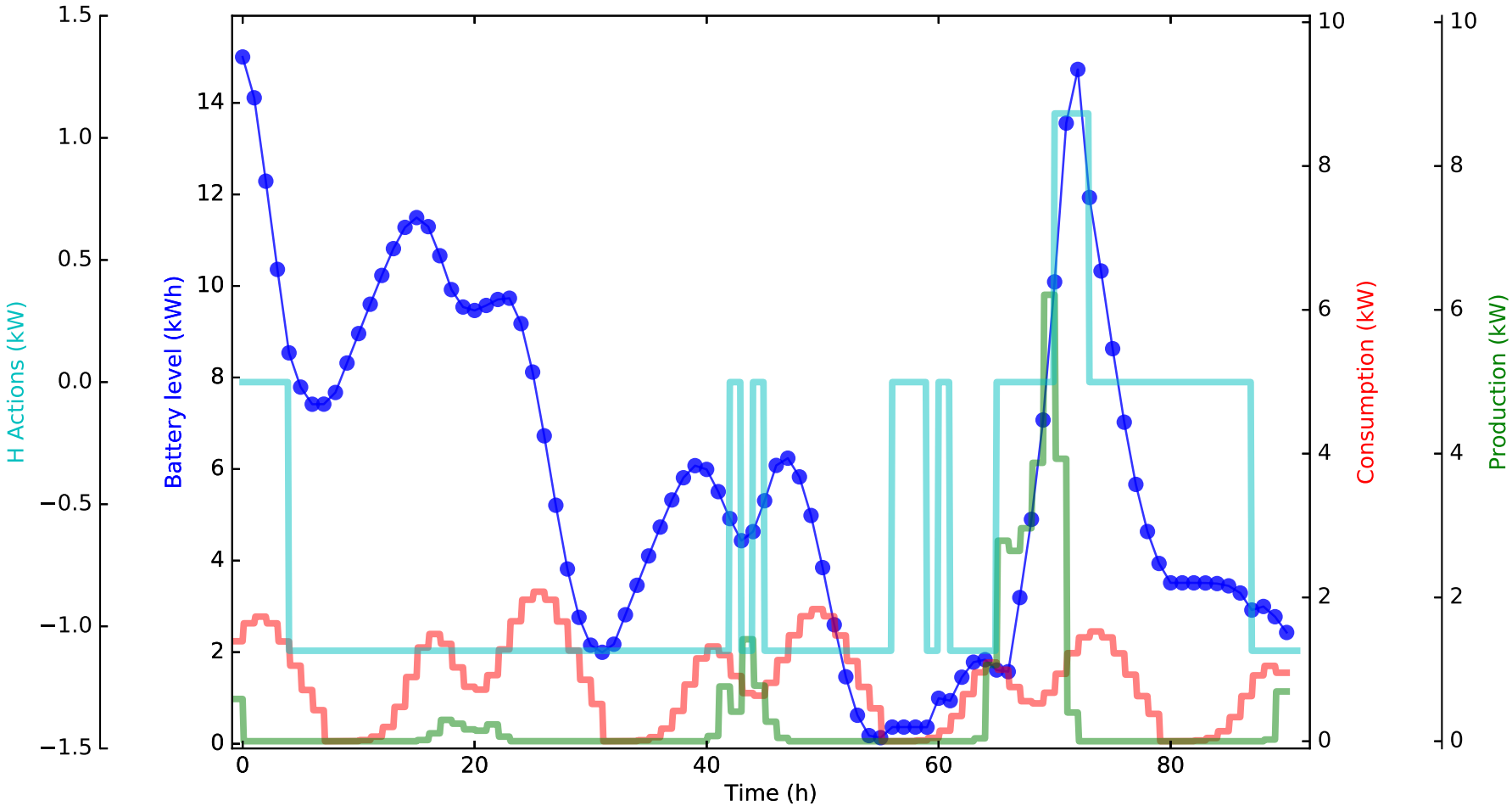
- past 12 hours for the production and consumption, and
- charge level of the battery.



Example

Illustration of the policy on the test data with the following features :

- past 12 hours for the production and consumption,
- level of the battery, and
- Accurate forecast for the mean production of the next 24h and 48h.



Results

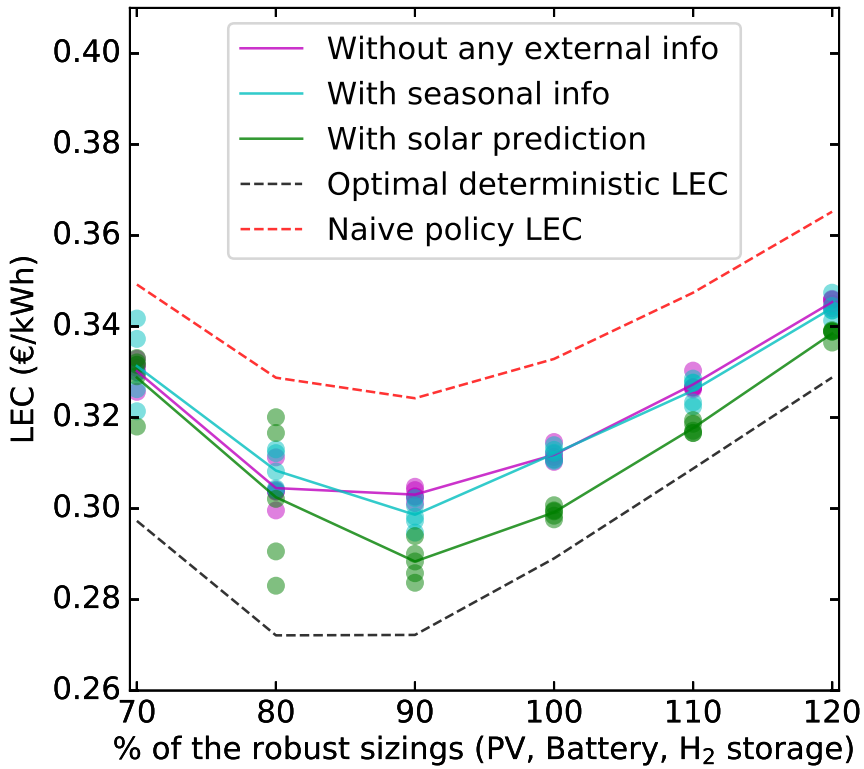


FIGURE – LEC on the test data function of the sizings of the microgrid.

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 : $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{P}$. 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.

Policy iteration algorithm

Policy iteration algorithm

1. Initialize policy π (e.g., randomly).
2. Compute value of policy V^{π}.
3. New policy π' to be a greedy policy with respect to V^{π} :

$$\pi' = \underset{a' \in \mathcal{A}}{\operatorname{argmax}} \left(r + \gamma \mathbb{E}_{s'} V^{\pi}(s') \right)$$

4. Repeat steps 2 and 3 until policy π does not change between two iterations.

NB : In deep RL, the policy π is parameterized by a neural network and the learning algorithms make use of the policy gradient theorems.

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

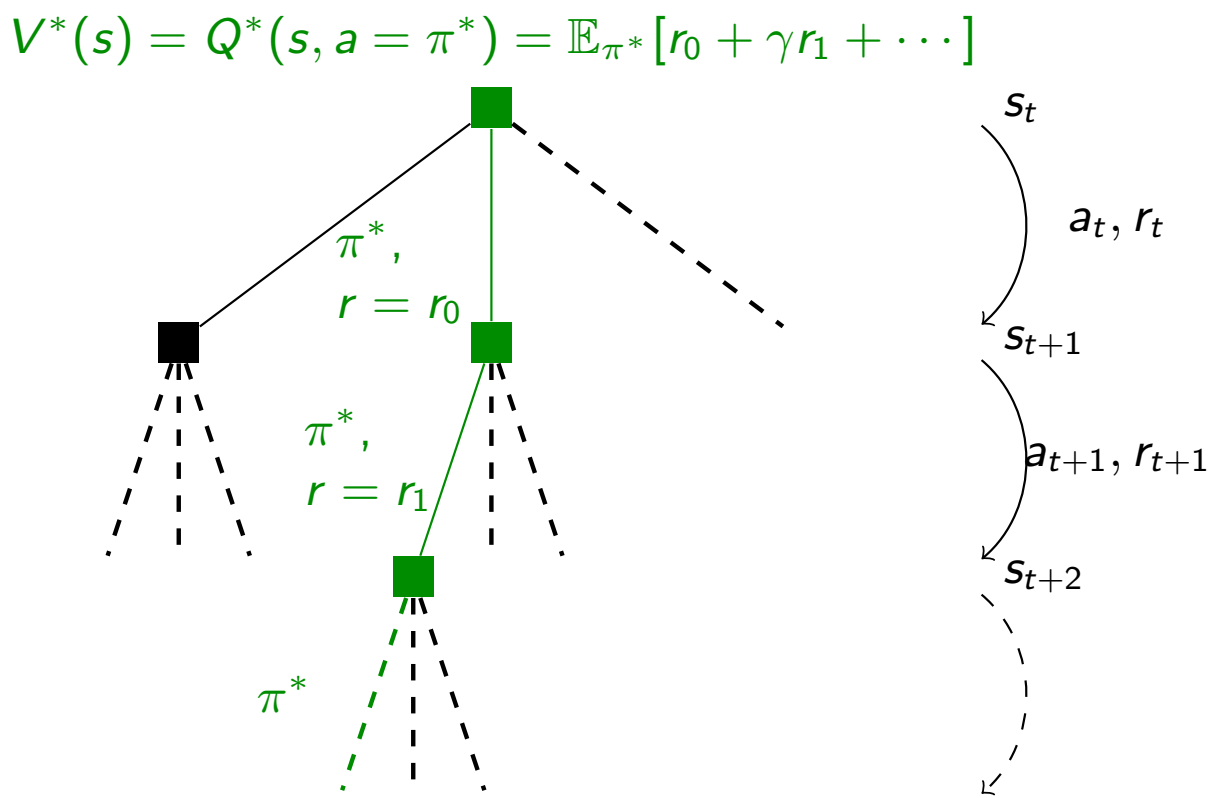
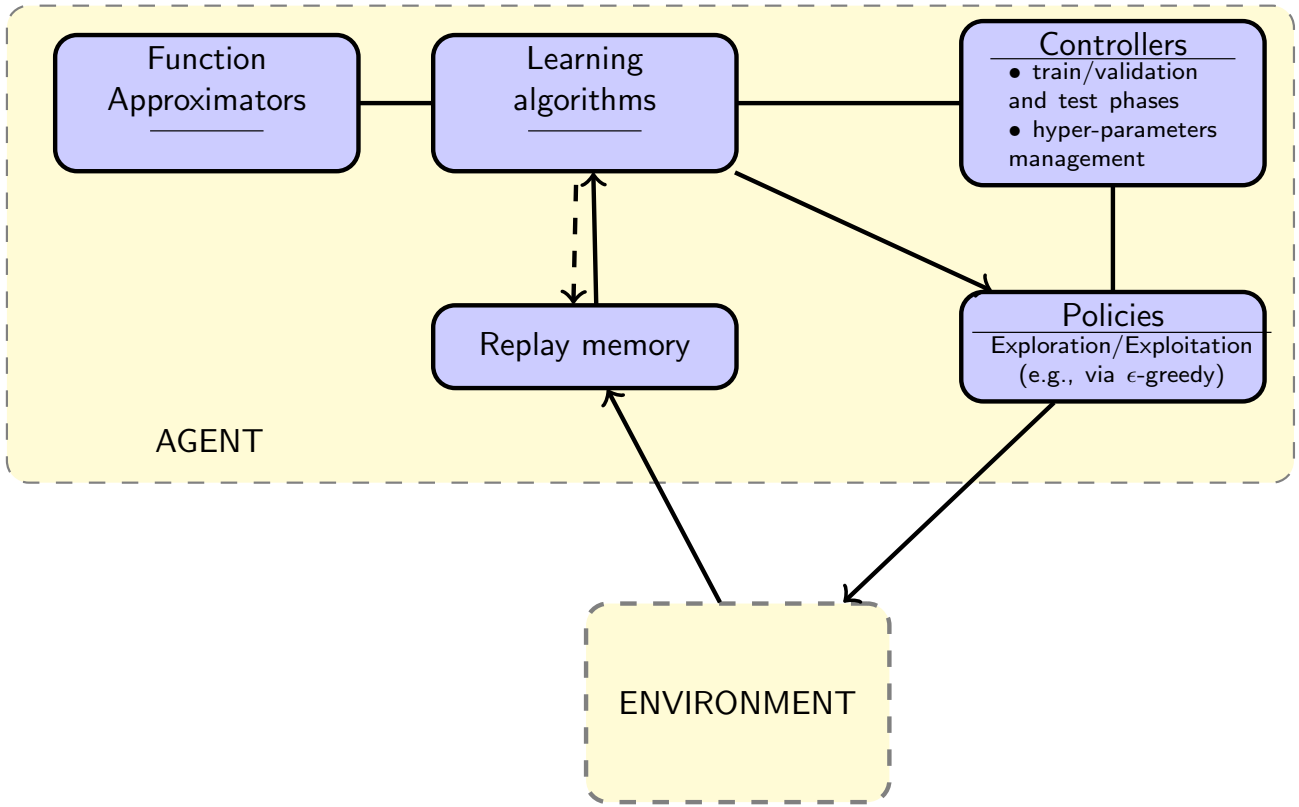


FIGURE – Illustration of model-based.

Overview

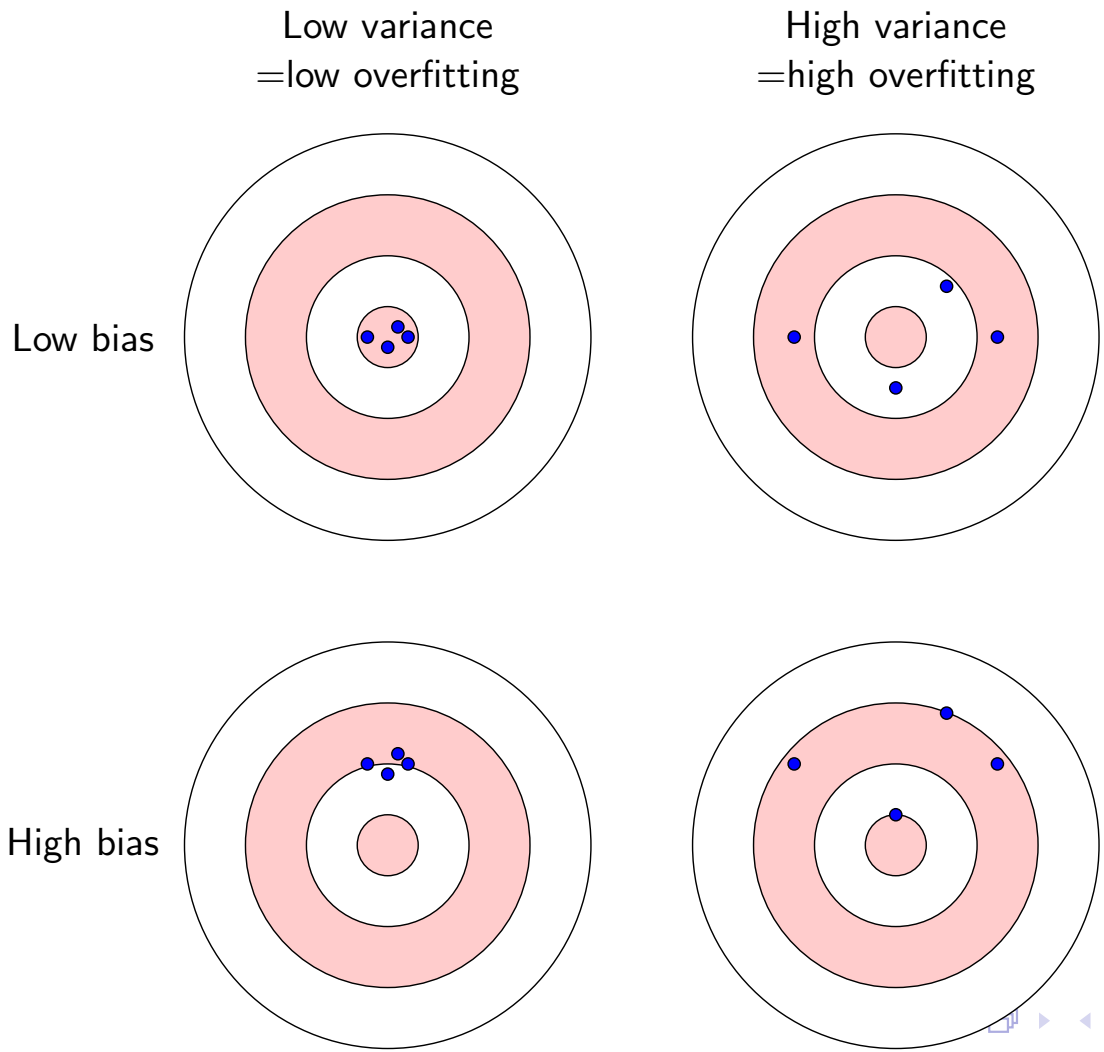


More details : V François-Lavet et al. *"An introduction to deep reinforcement learning"*. 140 pages. Foundations and Trends in ML.
Implementation : https ://github.com/VinF/deer

Generalisation from limited data

Bias and overfitting in supervised learning

A supervised learning algorithm can be viewed as a function that maps a dataset D_{LS} of learning samples $(x, y) \overset{\text{i.i.d.}}{\sim} (X, Y)$ into a predictive model $f(x \mid D_{LS})$.



Bias and overfitting in supervised learning

Assuming a random sampling scheme $D_{LS} \sim \mathcal{D}_{LS}$, $f(x \mid D_{LS})$ is a random variable, and so is its average error over the input space. The expected value of this quantity is given by :

$$I[f] = \mathop{\mathbb{E}}_{X} \mathop{\mathbb{E}}_{D_{LS}} \mathop{\mathbb{E}}_{Y|X} L(Y, f(X \mid D_{LS})), \tag{11}$$

where $L(\cdot, \cdot)$ is the loss function. If $L(y, \hat{y}) = (y - \hat{y})^2$, the error naturally gives the **bias-variance decomposition** :

$$\mathop{\mathbb{E}}_{D_{LS}} \mathop{\mathbb{E}}_{Y|X} (Y - f(X \mid D_{LS}))^2 = \sigma^2(x) + \text{bias}^2(x), \tag{12}$$

where

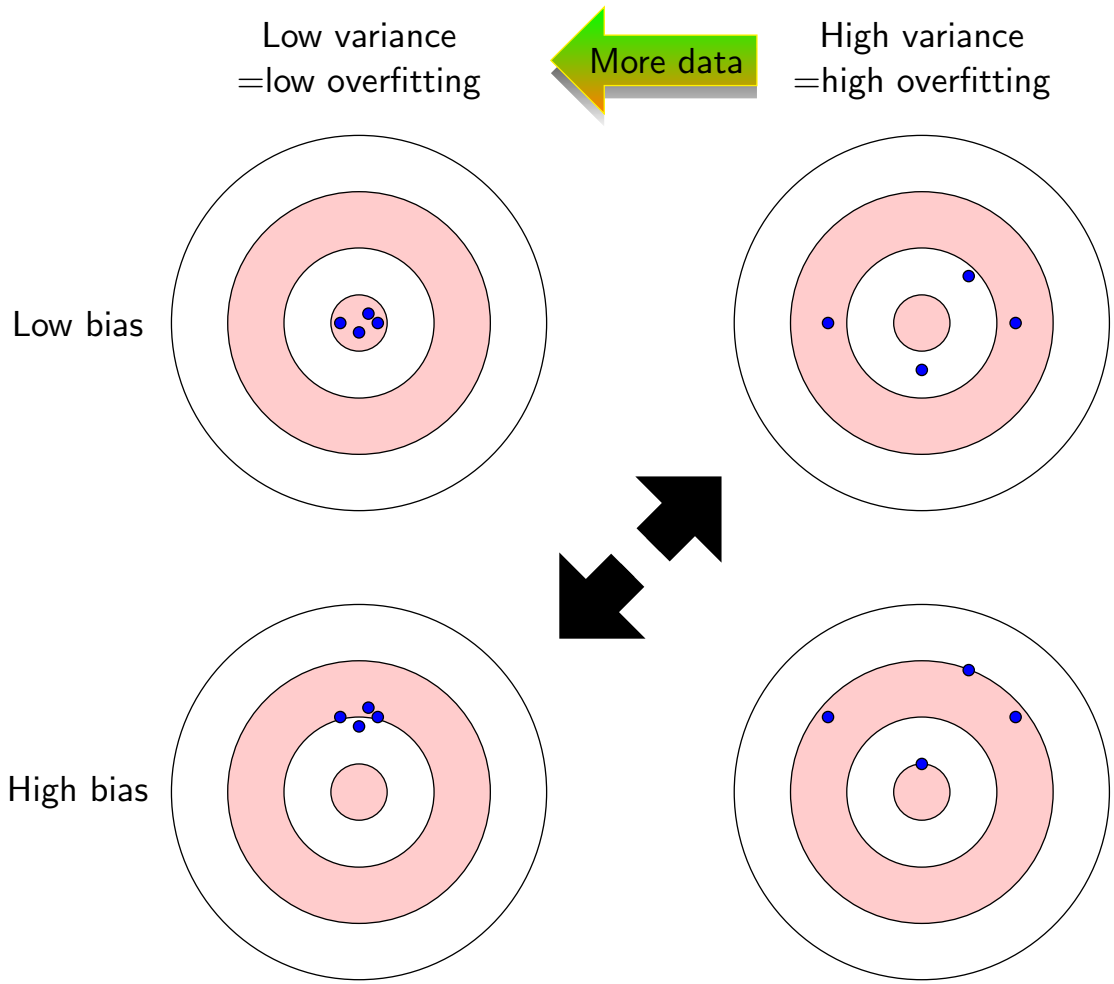
$$\text{bias}^2(x) \triangleq \left(\mathbb{E}_{Y|x}(Y) - \mathbb{E}_{D_{LS}} f(x \mid D_{LS})\right)^2,$$

$$\sigma^2(x) \triangleq \underbrace{\mathbb{E}_{Y|x}\left(Y - \mathbb{E}_{Y|x}(Y)\right)^2}_{\text{Internal variance}} + \underbrace{\mathbb{E}_{D_{LS}}\left(f(x \mid D_{LS}) - \mathbb{E}_{D_{LS}} f(x \mid D_{LS})\right)^2}_{\text{Parametric variance}}.$$

This bias-variance decomposition highlights a tradeoff between an error due to the learning algorithm (the bias) and an error due to the limited amount of data available (the parametric variance).

Bias and overfitting in supervised learning

There are many choices to optimize the learning algorithm and there is usually a tradeoff between the bias and the overfitting terms to reach to best solution.



Bias and overfitting in RL

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) :

$$D_s \rightarrow \pi_{D_s}.$$

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

$$\mathop{\mathbb{E}}_{D_s \sim \mathcal{D}_s}[V^{\pi^*}(s) - V^{\pi_{D_s}}(s)] = \underbrace{(V^{\pi^*}(s) - V^{\pi_{D_s,\infty}}(s))}_{\text{asymptotic bias}} + \underbrace{\mathop{\mathbb{E}}_{D_s \sim \mathcal{D}_s}[(V^{\pi_{D_s,\infty}}(s) - V^{\pi_{D_s}}(s))\ ]}_{\substack{\text{error due to finite size of the dataset } D_s \\ \text{referred to as overfitting}}}. \quad (13)$$

How to obtain the best policy ?

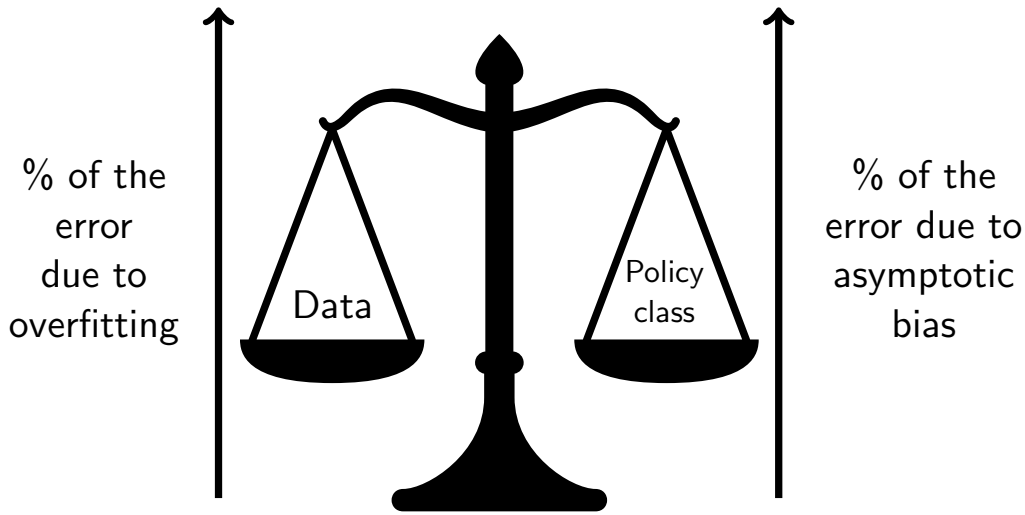


FIGURE – Schematic representation of the bias-overfitting tradeoff.

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)

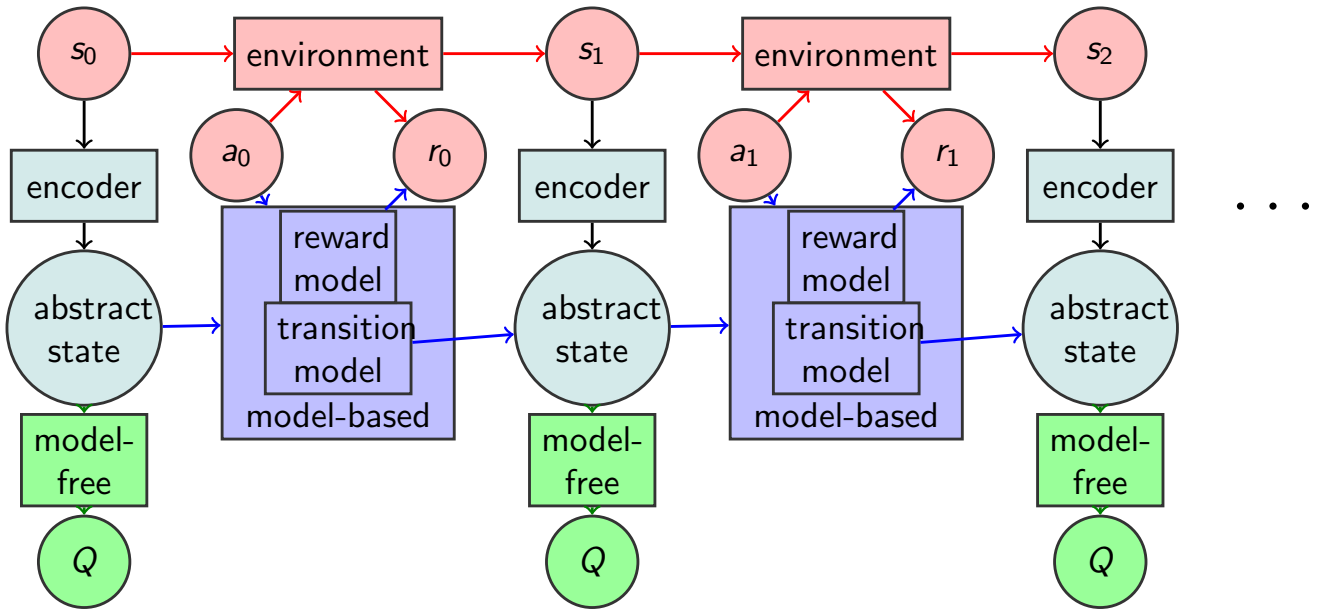


FIGURE – Illustration of the integration of model-based and model-free RL in the CRAR architecture, with a low-dimensional abstract state over which transitions and rewards are modeled.

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

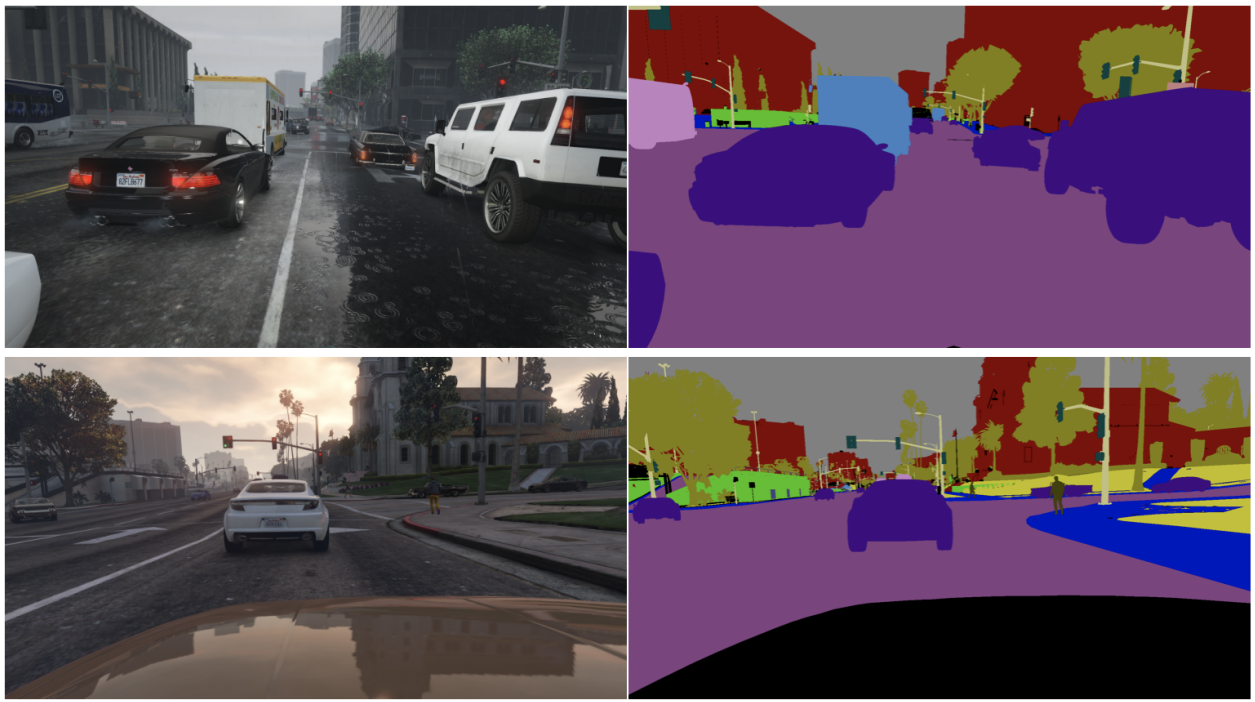


FIGURE – Transfer learning between different renderings. Picture from *"Playing for Data : Ground Truth from Computer Games", Richter, S. and Vineet, V., et al*

Transfer learning with the CRAR agent

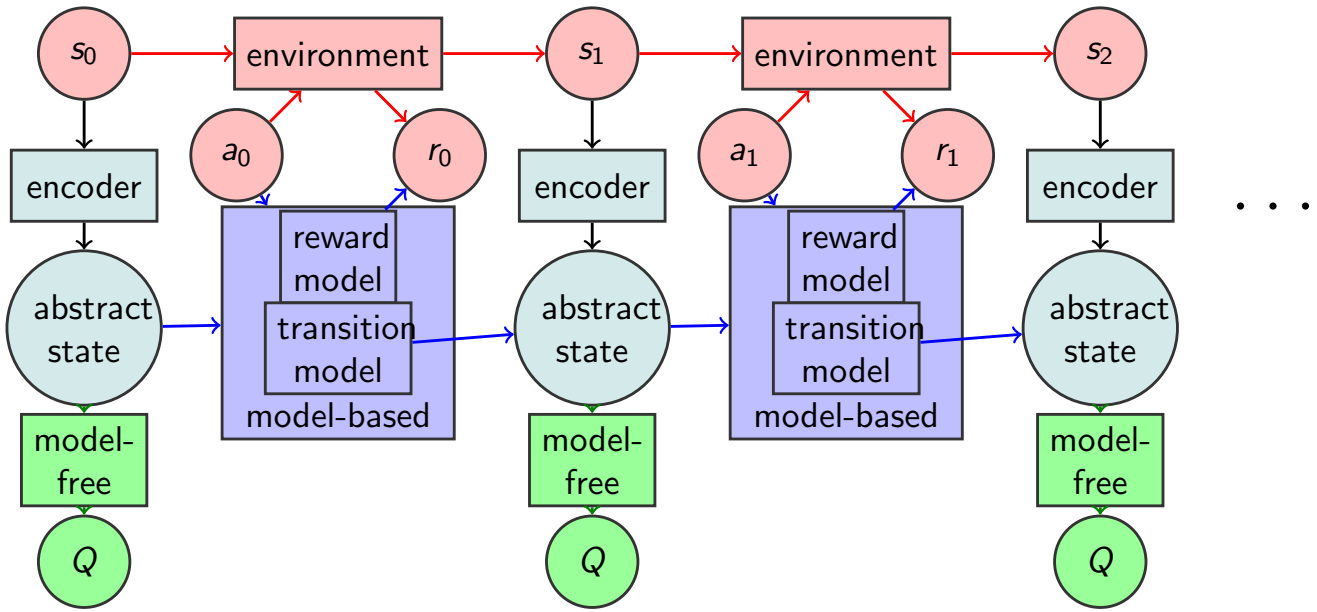


FIGURE – Illustration of the integration of model-based and model-free RL in the CRAR architecture, with a low-dimensional abstract state over which transitions and rewards are modeled.

Discussion of a 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)

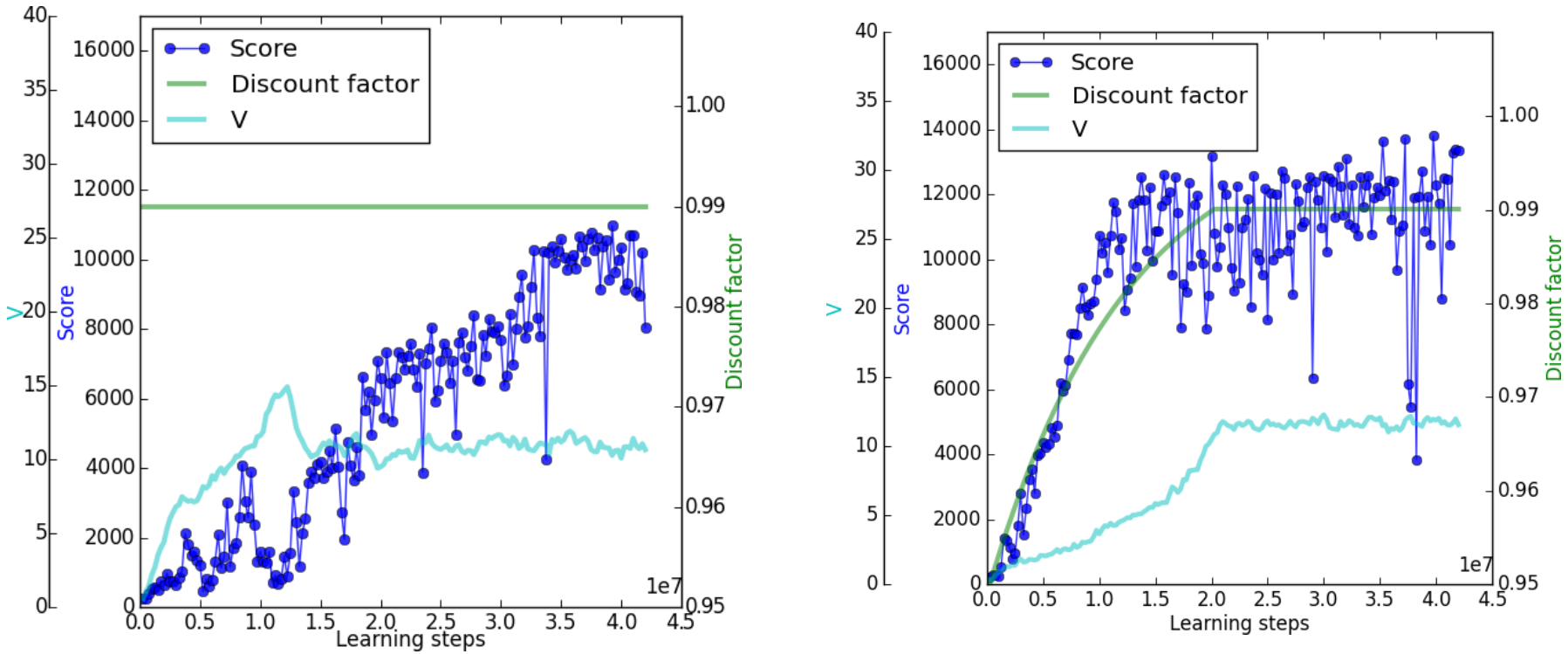


FIGURE – Illustration for the game q-bert of a discount factor γ held fixed on the right and an adaptive discount factor on the right.

Further ressources

- Richard Sutton 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
- Vincent François-Lavet et al. *"An introduction to deep reinforcement learning"*. Foundations and Trends in ML.

Conclusions

Summary

- Introduction to reinforcement learning and deep reinforcement learning
- The importance of generalization with limited data
- Why combining model-free and model-based approaches
- Brief discussion on some relations to neuroscience

Questions ?