Kickoff RL week + generalization in reinforcement learning

Vincent François-Lavet

July 11th, 2022

Outline

Kickoff RL week

Motivation for generalization and deep reinforcement learning

Generalisation from limited data in supervised learning

Generalisation from limited data in reinforcement learning

How to improve generalization in RL?

A few additional challenges in (deep) RL

Conclusions

Kickoff RL week

Unfolding of the week

A program meant to

- getting up to speed with the fundamentals of RL (particularly the first few days),
- learning some of the latest developments in RL (particularly the last few days), and
- connecting with each other (social activity, practical sessions, final quizz).

We wanted to keep this RL summer school at a reasonable size to allow for interactions, please do not hesitate to ask questions (with moderation for each student ;)

Some numbers

- About **80 participants** from all continents (**17 different countries**).
- **12 speakers**, **5 teaching assistants**.
- **5 days**

If you want to connect with each other during the week, you can join this what's app group (you are already about 50 to have joined)
https://chat.whatsapp.com/FnOwY1d7WW8JoyESQMUzrz

Social activity - Wednesday afternoon

It's a boat tour with Italian lunch (vegan/vegetarian option should be available).

Information to get there: You can take tram number 5 to Rijksmuseum walk underneath and on the other side there is a pick up place on the left side of the bridge. We are planning to get onboard at **1:45pm**.

Additional information

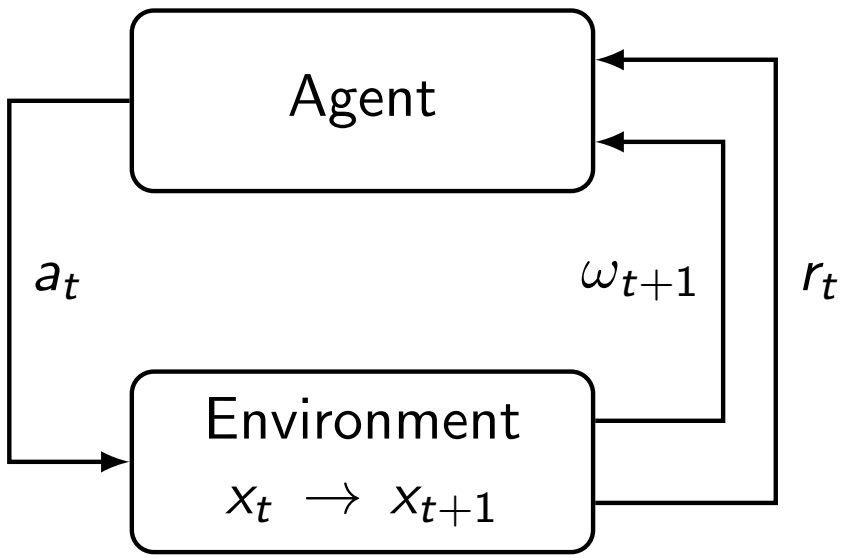
There should be (if everything goes fine) free lunch every day and we'll try to organise also a (free) pizza dinner on Friday after the last session.

In order to be as efficient as possible, it is advised that you already check https://github.com/VinF/practical_sessions_RL and follow the instruction in the README such that you have everything ready for the practical sessions.

Motivation for generalization and deep reinforcement learning

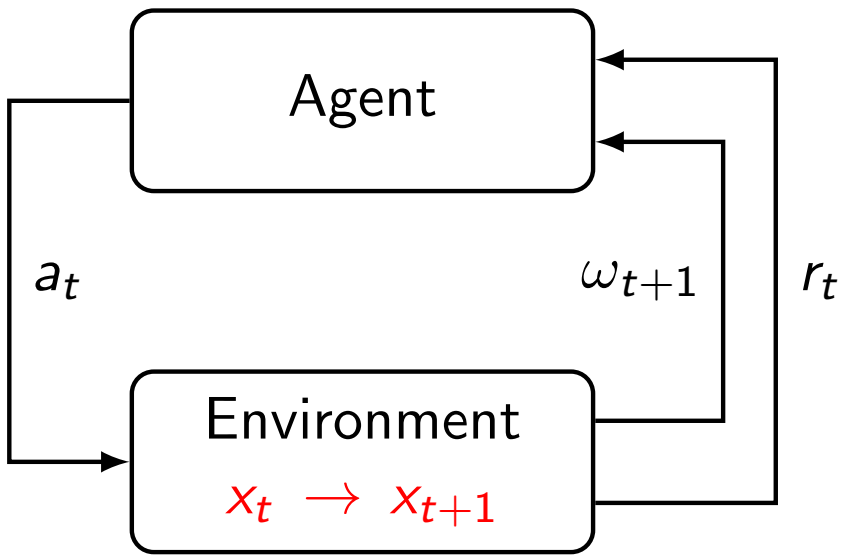
Objective

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



Objective

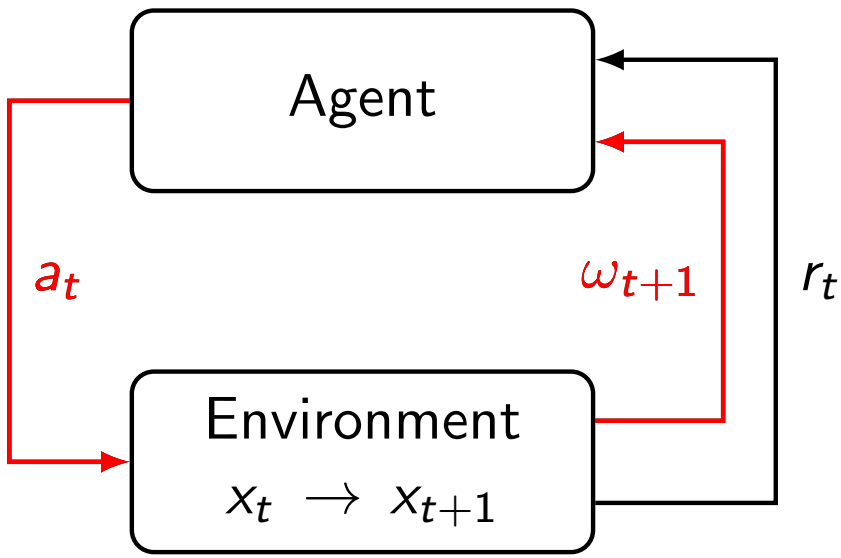
From experience in an environment,
an artificial agent
should be able to **learn** a sequential decision making task
in order **to achieve goals**.



transitions
are usually
stochastic

Objective

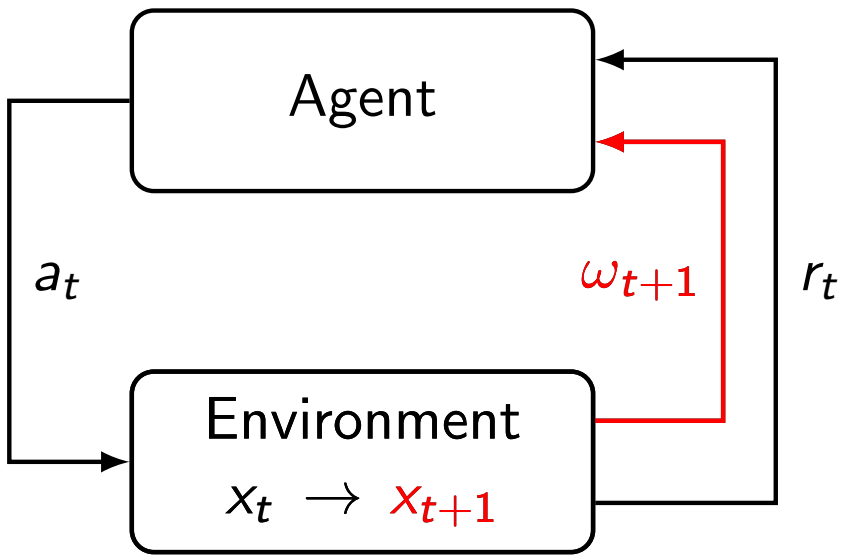
From experience in an environment,
an artificial agent
should be able to **learn** a sequential decision making task
in order **to achieve goals**.



Observations and
actions may be
high dimensional

Objective

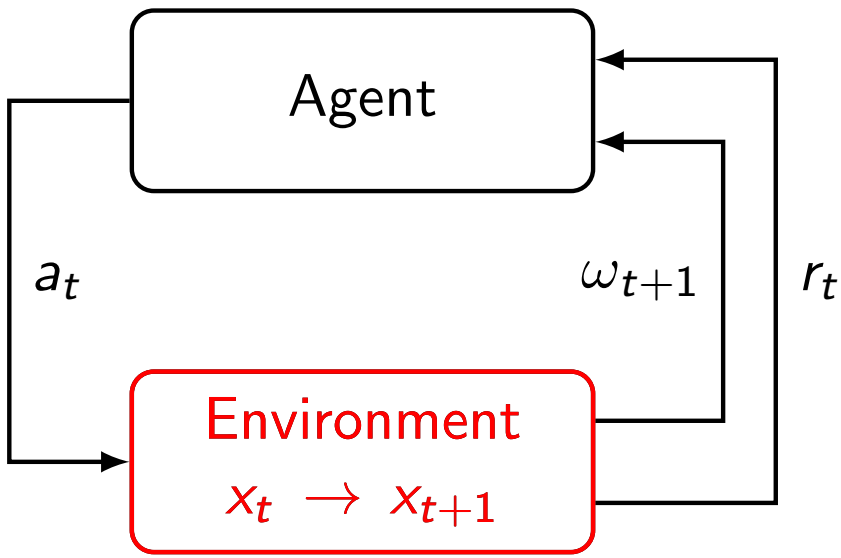
From experience in an environment,
an artificial agent
should be able to **learn** a sequential decision making task
in order **to achieve goals**.



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

Objective

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

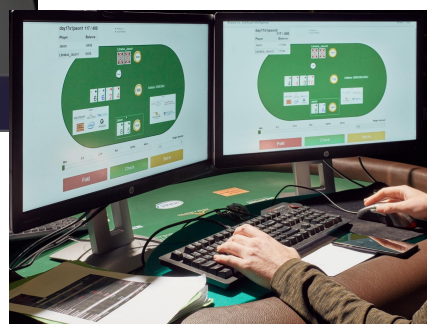


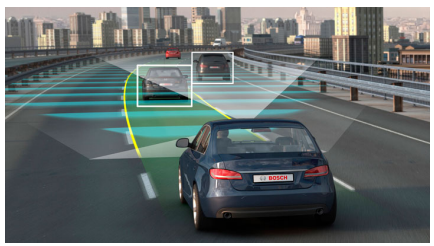
Experience may be constrained
(e.g., not access to an accurate simulator or limited data)

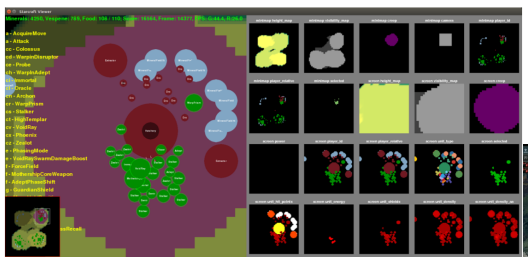
Motivation

Figure: Example of an ATARI game: Seaquest

Motivation: Overview







Challenges of applying RL to real-world problems

In real-world scenarios, it is often not possible to let an agent interact freely and sufficiently in the actual environment:

1. The agent may not be able to interact with the true environment but only with an inaccurate simulation of it. This is known as the **reality gap**.
2. The agent might have access to only **limited data**. This can be due to safety constraints (robotics, medical trials, etc.), compute constraints or due to limited exogenous data (e.g., weather conditions, trading markets).

Challenges of applying RL to real-world problems

In order to deal with the reality gap and limited data, different elements are important:

- One can aim to develop **a simulator that is as accurate as possible**.
- One can design the learning algorithm so as to **improve generalization** (and/or use specific transfer learning methods).

Generalization

In an RL algorithm, *generalization* refers to either

- the capacity to achieve good performance in an environment where limited data has been gathered, or
- the capacity to obtain good performance in a related environment. This latter case can be tackled with specific *transfer learning* techniques.

Overview

To understand generalization in RL from limited data, we will

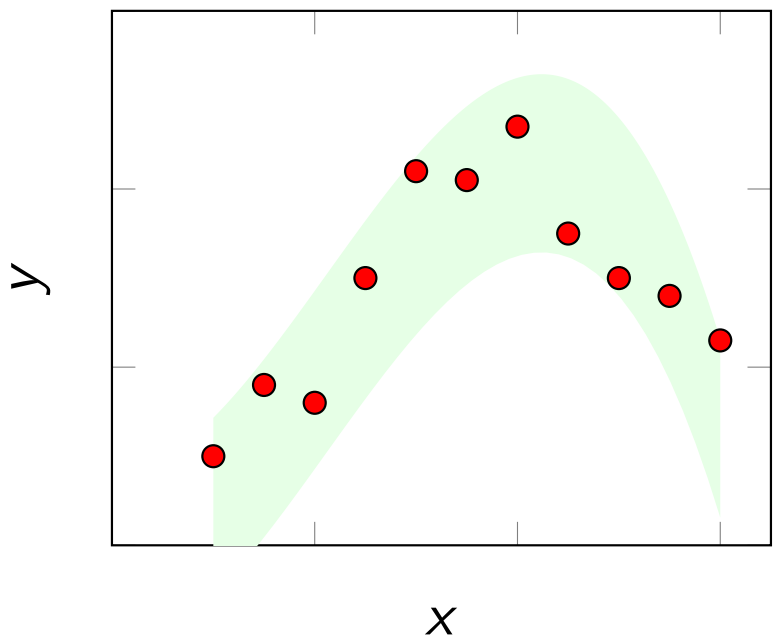
- recall the concept in supervised learning, and
- introduce the formulation in RL.

We'll then discuss how an agent can have a good generalization in RL (disclaimer: we'll see where deep RL comes in !)

Generalisation from limited data in supervised learning

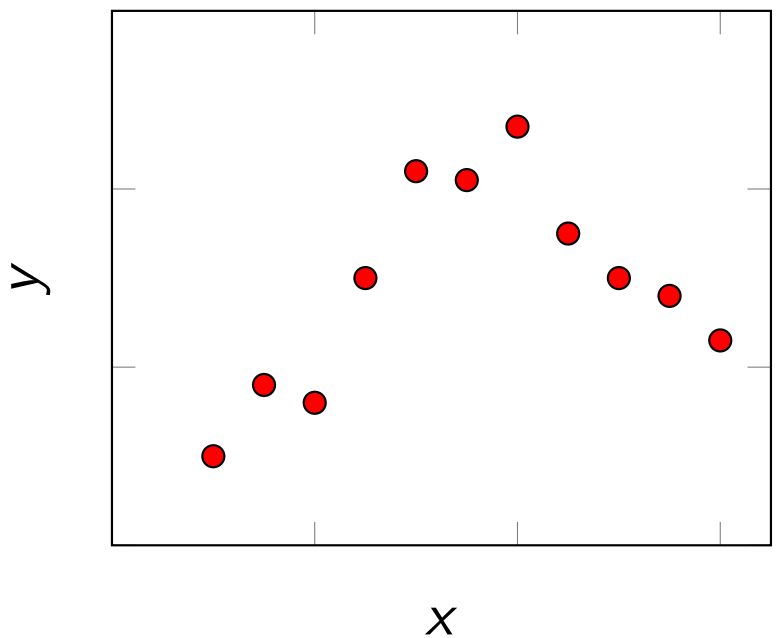
Bias and overfitting in supervised learning

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



Bias and overfitting in supervised learning

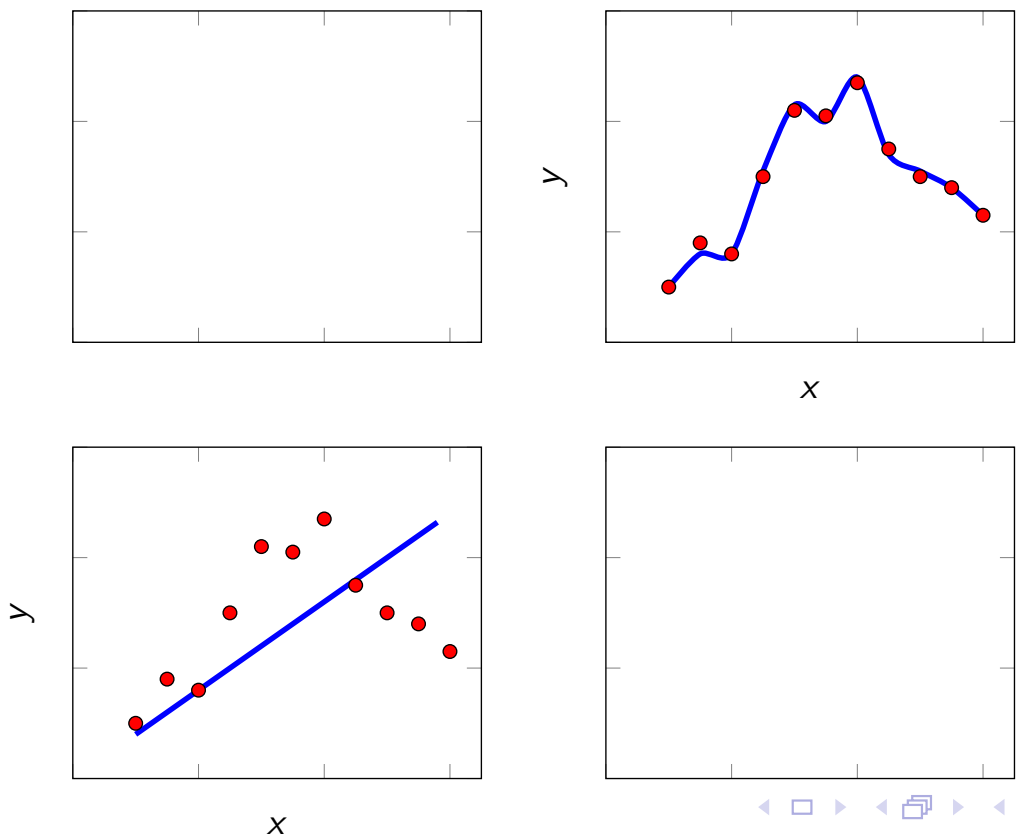
A supervised learning algorithm can be viewed as a mapping from a dataset D_{LS} of learning samples $(x, y) \overset{\text{i.i.d.}}{\sim} (X, Y)$ into a predictive model.



You only have access to a limited dataset.

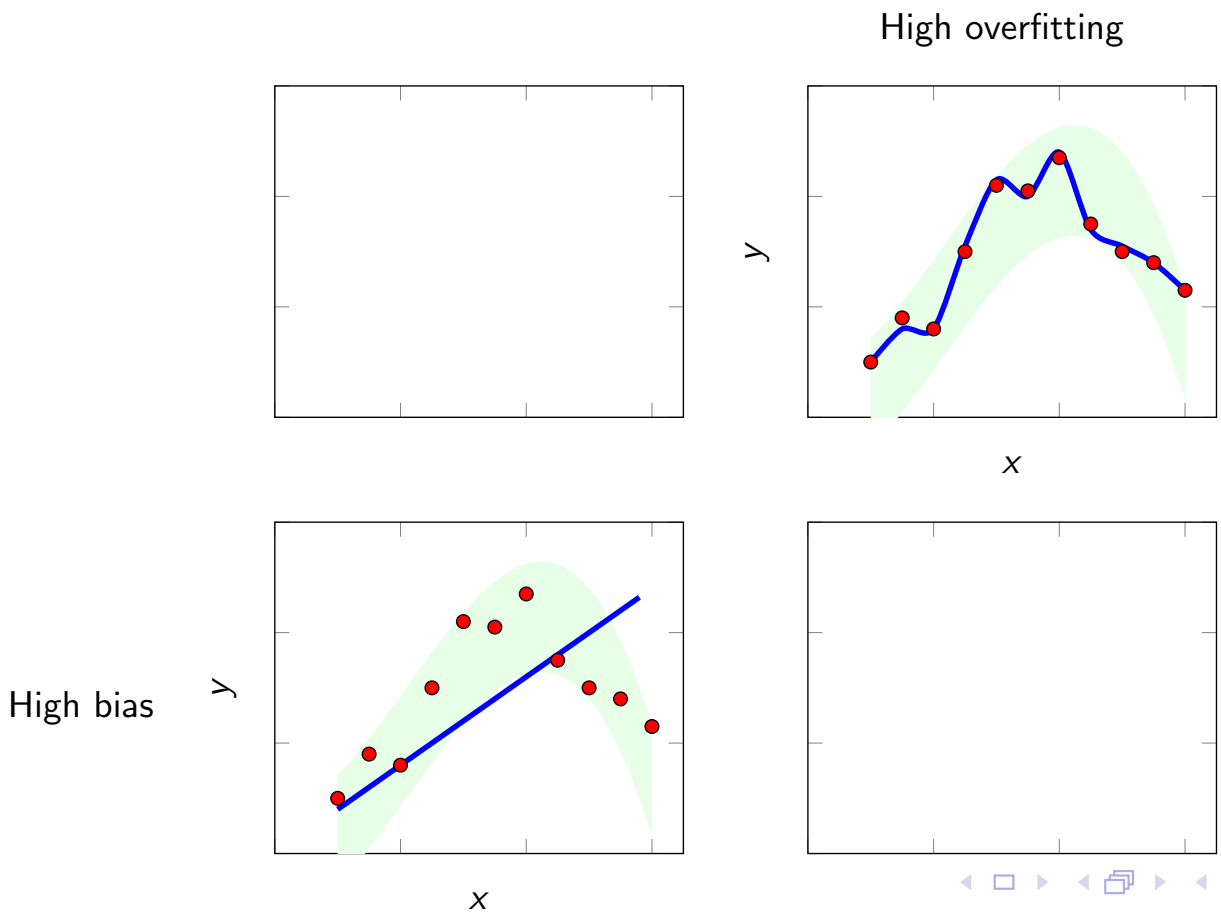
Bias and overfitting in supervised learning

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



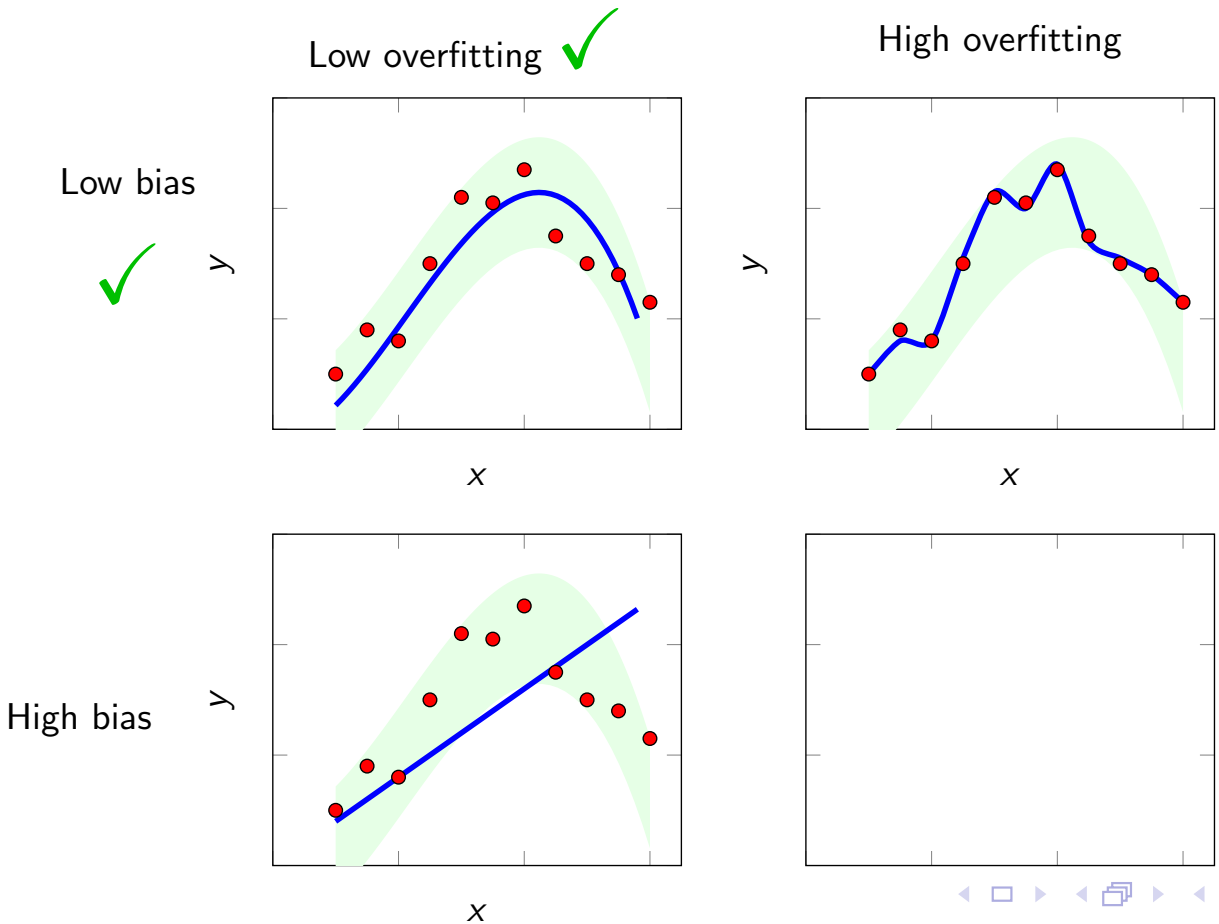
Bias and overfitting in supervised learning

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



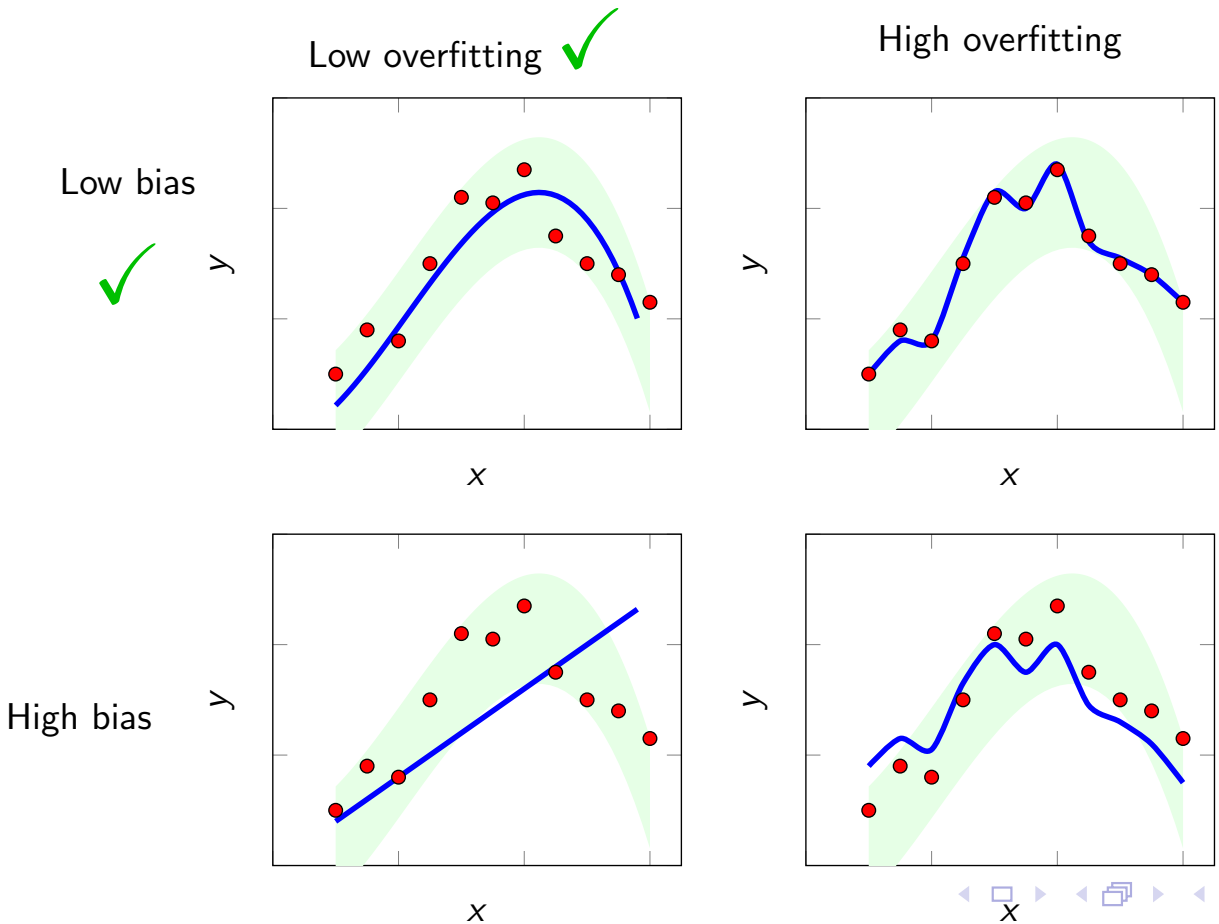
Bias and overfitting in supervised learning

A supervised learning algorithm can be viewed as a mapping from 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

A supervised learning algorithm can be viewed as a mapping from 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

For one given $x \sim X$, the predictive model $f(x \mid D_{LS})$ can be illustrated as follows for unseen data $y \sim (Y \mid X = x)$:

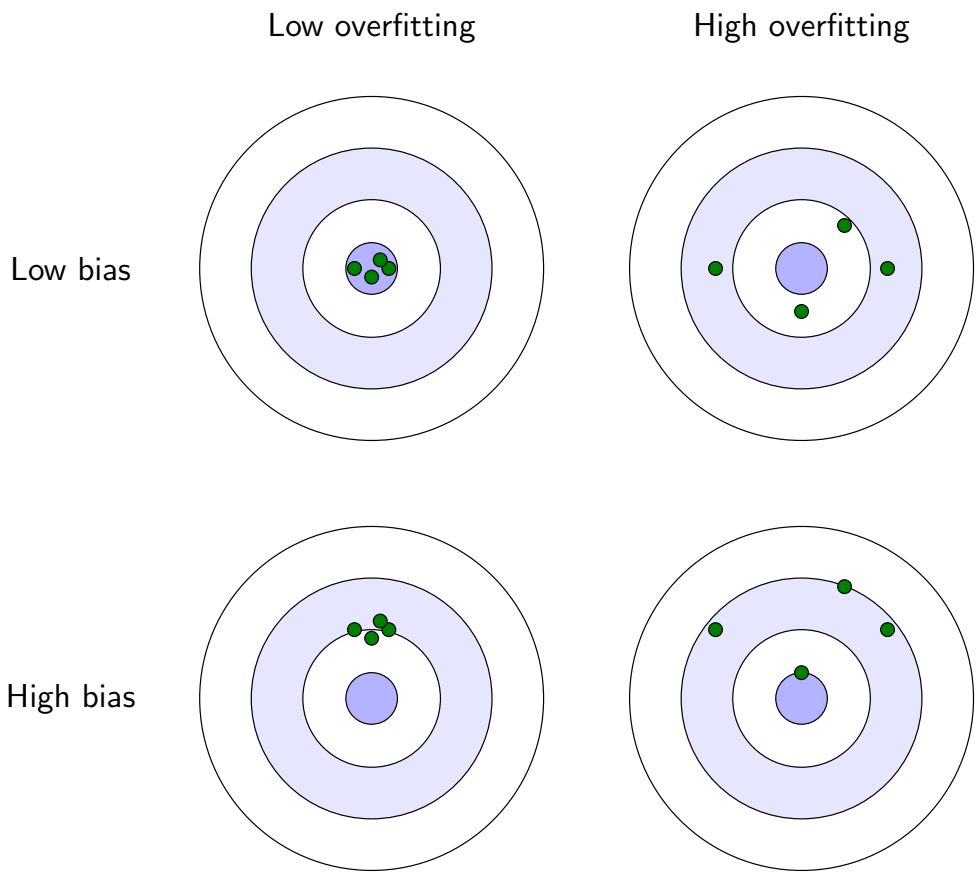
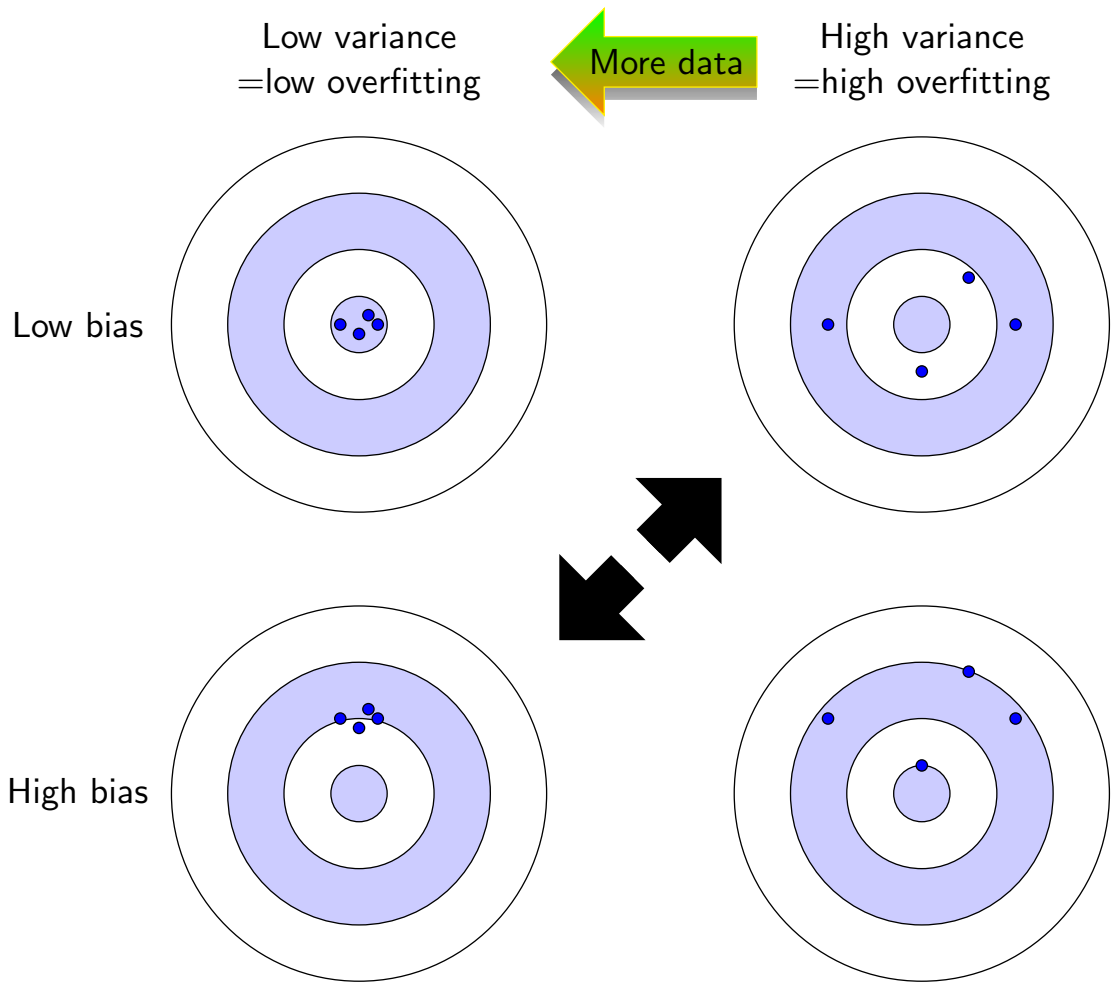


Figure: Illustration of bias and overfitting for unseen tuples, where Y is a 2D continuous RV for visualisation purposes.

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 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:

$$I[f] = \mathop{\mathbb{E}}_{X} \mathop{\mathbb{E}}_{D_{LS}} \mathop{\mathbb{E}}_{Y|X} L(Y, f(X \mid D_{LS})), \tag{1}$$

where $L(\cdot, \cdot)$ is the loss function.

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:

$$I[f] = \mathop{\mathbb{E}}_{X} \mathop{\mathbb{E}}_{D_{LS}} \mathop{\mathbb{E}}_{Y|X} L(Y, f(X \mid D_{LS})), \tag{1}$$

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**:

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

where

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

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

Bias and overfitting in reinforcement learning

This bias-variance decomposition highlights a tradeoff between

- an error directly introduced by the learning algorithm (the bias) and
- an error due to the limited amount of data available (the parametric variance).

Bias and overfitting in reinforcement learning

This bias-variance decomposition highlights a tradeoff between

- an error directly introduced by the learning algorithm (the bias) and
- an error due to the limited amount of data available (the parametric variance).

Note that there is no such direct bias-variance decomposition for loss functions other than the L_2 loss! It is however always possible to decompose the prediction error with a term related to the lack of expressivity of the model (the bias) and a term due to the limited amount of data (overfitting comes from the variance of $f(x \mid D_{LS})$ on the loss when $D_{LS} \sim \mathcal{D}_{LS}$ but $\neq$ statistical variance if loss function is not L_2).

Break

Next: generalization in reinforcement learning

Generalisation from limited data in reinforcement learning

Bias and overfitting in supervised learning

Since there is no direct bias-variance decomposition for loss functions other than L_2 loss in supervised learning, there is not an actual "bias-variance" tradeoff in RL.

However, there is still a tradeoff between a sufficiently rich learning algorithm (to reduce the model bias, which is present even when the amount of data would be unlimited) and a learning algorithm not too complex (so as to avoid overfitting to the limited amount of data).

Bias and overfitting in RL

The *batch* or *offline* algorithm in RL can be seen as mapping a dataset $D \sim \mathcal{D}$ into a policy π_D (independently of whether the policy comes from a model-based or a model-free approach):

$$D \rightarrow \pi_D.$$

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

$$\mathop{\mathbb{E}}_{D\sim\mathcal{D}}[V^{\pi^*}(x) - V^{\pi_D}(x)] = \underbrace{(V^{\pi^*}(x) - V^{\pi_{D_\infty}}(x))}_{\textit{asymptotic bias}} + \underbrace{\mathop{\mathbb{E}}_{D\sim\mathcal{D}}[(V^{\pi_{D_\infty}}(x) - V^{\pi_D}(x)) \quad]}_{\begin{array}{c}\text{error due to finite size of the dataset } D_s \\ \text{referred to as } \textit{overfitting}\end{array}}.$$

How to obtain the best policy?

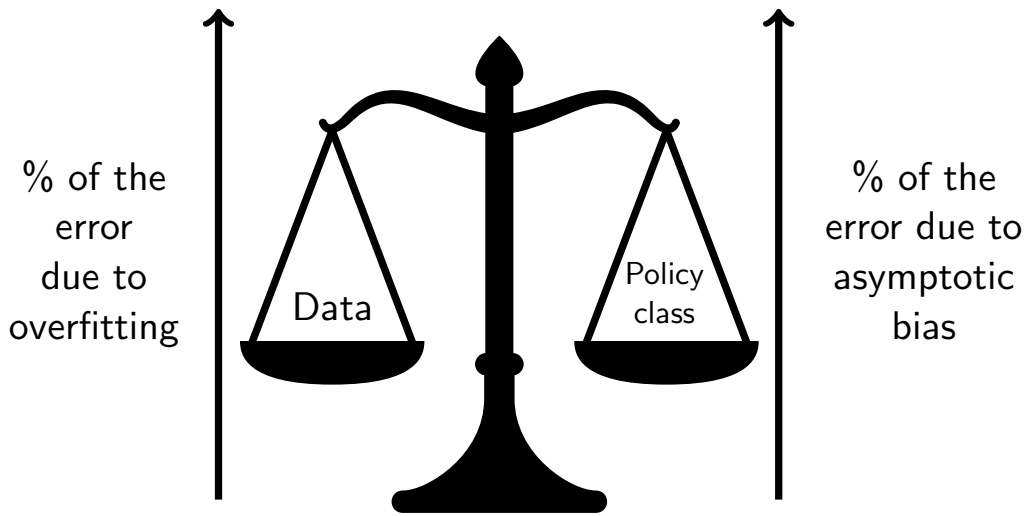


Figure: Schematic representation of the bias-overfitting tradeoff.

How to improve generalization in RL?

How to improve generalization?

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

- an **abstract representation** that discards non-essential features,
- 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)

1. Abstract representation

The appropriate level of abstraction plays a key role in the bias-overfitting tradeoff and one of the key advantages of using a small but rich abstract representation is to allow for improved generalization.

- When considering many features on which to base the policy, an RL algorithm may take into consideration spurious correlations, which leads to overfitting.
- Removing features that discriminate states with a very different role in the dynamics introduces a bias.

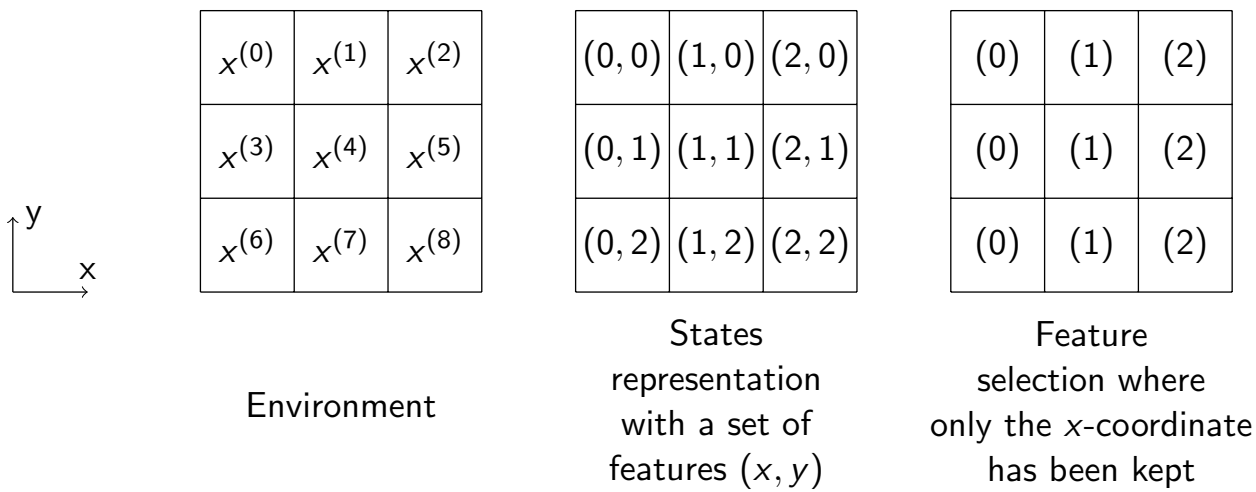


Figure: Illustration of the abstract representation.

2. Modifying the objective function

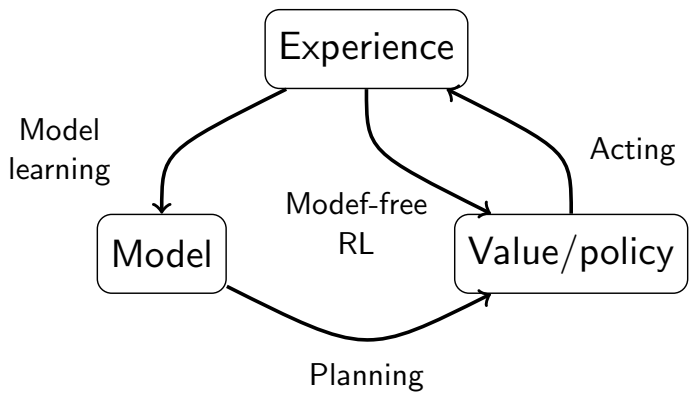
In order to improve the policy learned by a deep RL algorithm, one can optimize an objective function that diverts from the actual objective. By doing so, a bias is usually introduced but this can in some cases help with generalization. The main approaches to modify the objective function are either

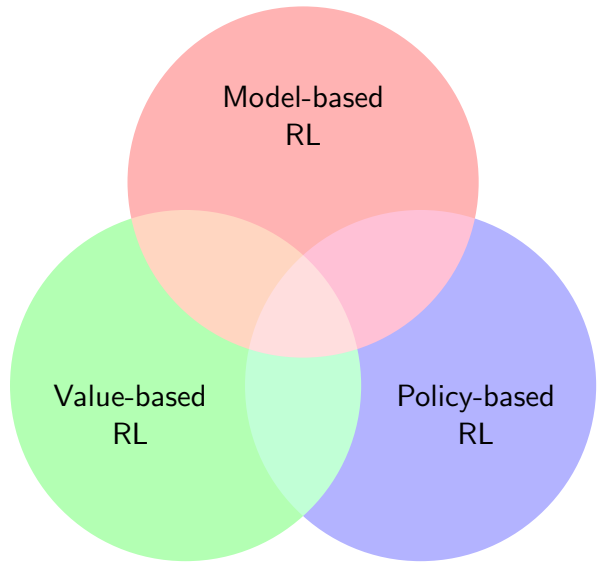
- to modify the reward of the task to ease learning (reward shaping), or
- tune the discount factor at training time.

3. Choice of the learning algorithm and function approximator selection

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 $\pi(x)$ or $\pi(x, a)$, or
- a model of the environment in conjunction with a planning algorithm.





Deep learning has brought its generalization capabilities to RL.

3. Choice of the learning algorithm and function approximator selection

- The function approximator in deep learning characterizes how the features will be treated into higher levels of abstraction. A fortiori, it is related to feature selections (e.g., an attention mechanism), etc.
- Depending on the task, finding a performant function approximator is easier in either a model-free or a model-based approach. The choice of relying more on one or the other approach is thus also a crucial element to improve generalization.

3. Choice of the learning algorithm: a parallel with neurosciences

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.

A few additional challenges in (deep) RL

Another important challenge: transfer learning

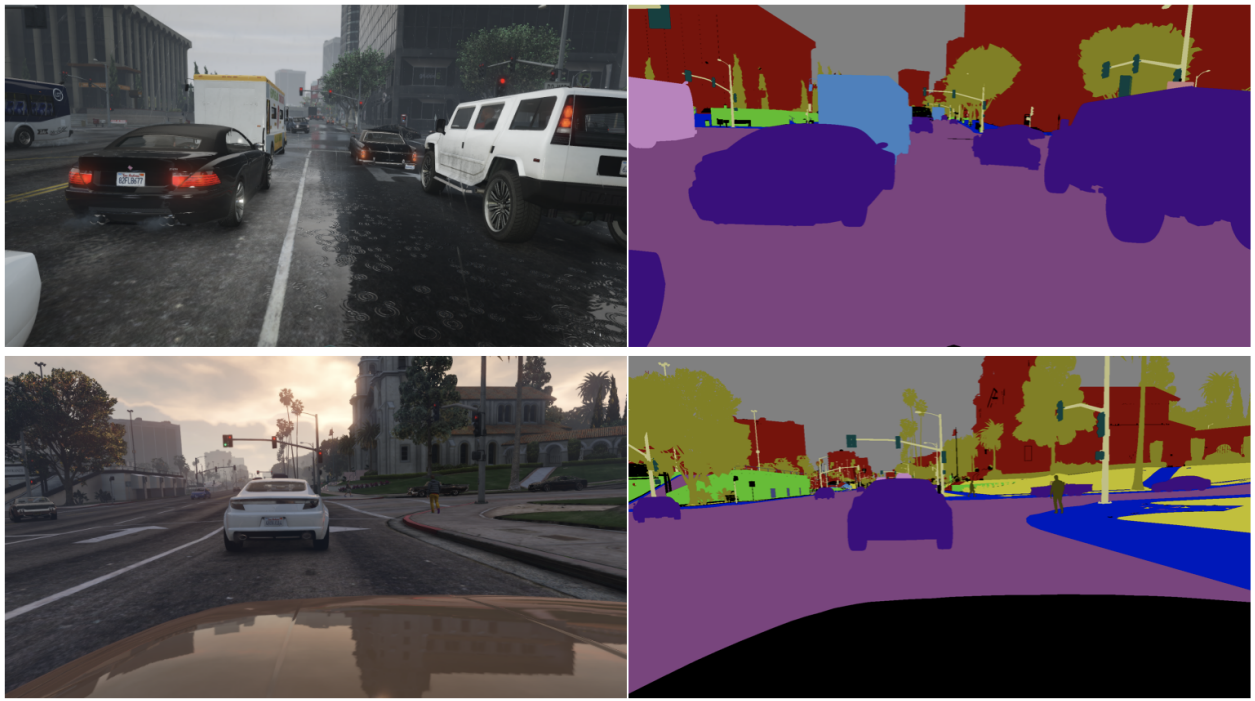


Figure: Transfer learning between different renderings. Picture from *"Playing for Data: Ground Truth from Computer Games", Richter, S. and Vineet, V., et al*

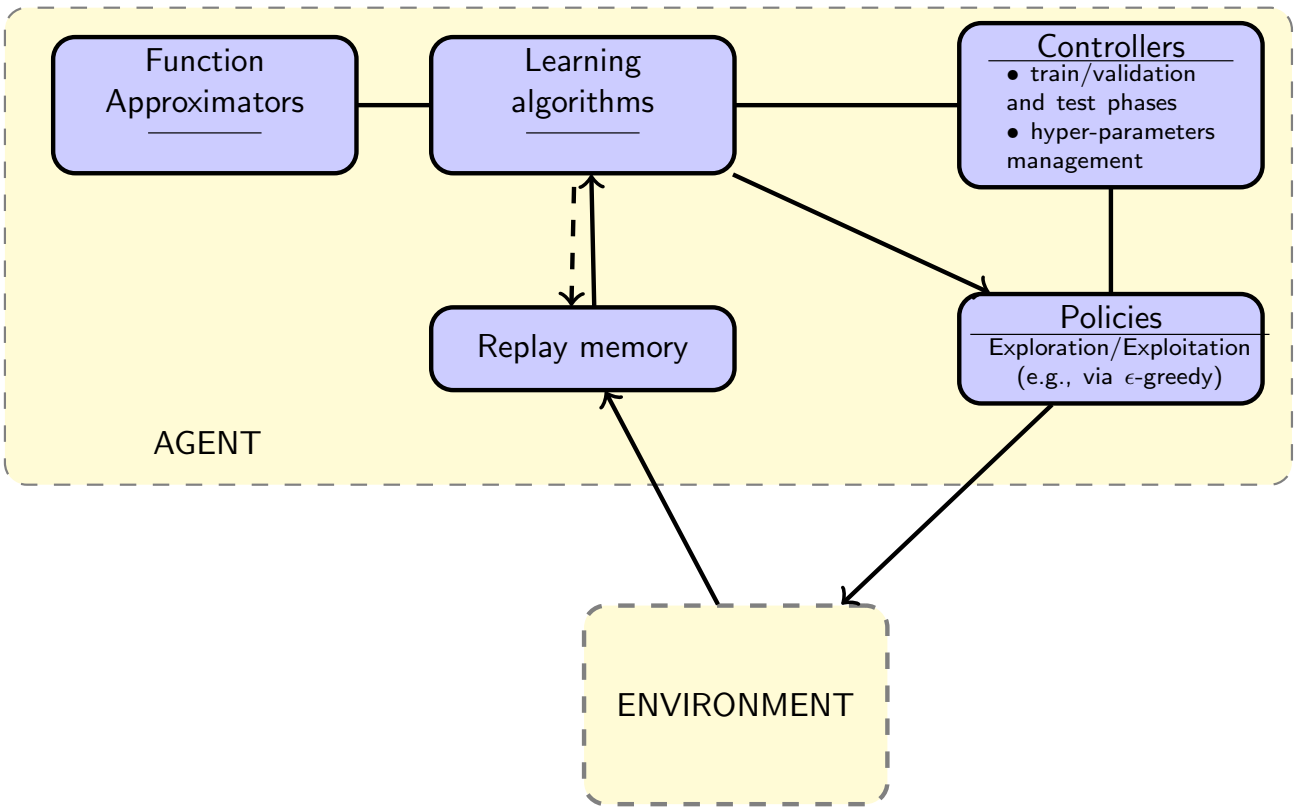
Another important challenge: exploration

- Undirected exploration (e.g. ϵ-greedy)
- Directed exploration with an estimate of "novelty"
  - when rewards are not sparse, a measure of the uncertainty on the value function can be used,
  - if sparse rewards or no rewards, some exploration rewards have to be used.

Conclusions

Conclusion

The different components of the RL agent play a role in performance.



Conclusion

Generalization is a central concept in the field of machine learning, and reinforcement learning is no exception.

Today, we have seen

- what generalization is in RL,
- how an agent can have a good generalization,
- a few additional challenges in (deep) RL

More information can be found in the following book:

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

Questions?