Computational Economics
https://doi.org/10.1007/s10614-021-10119-4

Reinforcement Learning in Economics and Finance

Arthur Charpentier[1] · Romuald Élie[2] · Carl Remlinger[2]

Accepted: 7 April 2021

Abstract

Reinforcement learning algorithms describe how an agent can learn an optimal action policy in a sequential decision process, through repeated experience. In a given environment, the agent policy provides him some running and terminal rewards. As in online learning, the agent learns sequentially. As in multi-armed bandit problems, when an agent picks an action, he can not infer ex-post the rewards induced by other action choices. In reinforcement learning, his actions have consequences: they influence not only rewards, but also future states of the world. The goal of reinforcement learning is to find an optimal policy – a mapping from the states of the world to the set of actions, in order to maximize cumulative reward, which is a long term strategy. Exploring might be sub-optimal on a short-term horizon but could lead to optimal long-term ones. Many problems of optimal control, popular in economics for more than forty years, can be expressed in the reinforcement learning framework, and recent advances in computational science, provided in particular by deep learning algorithms, can be used by economists in order to solve complex behavioral problems. In this article, we propose a state-of-the-art of reinforcement learning techniques, and present applications in economics, game theory, operation research and finance.

Keywords Causality · Control · Machine learning · Markov decision process · Multi-armed bandits · Online-learning · *Q*-learning · Regret · Reinforcement learning · Rewards · Sequential learning

Arthur Charpentier acknowledges the financial support of the AXA Research Fund through the joint research initiative Use and value of unusual data in actuarial science, as well as NSERC grant 2019-07077.

✉ Arthur Charpentier
arthur.charpentier@uqam.ca

Romuald Élie
romuald.elie@u-pem.fr

[1] Université du Québec à Montréal (UQAM), 201, avenue du Président-Kennedy, Montréal (Québec) H2X 3Y7, Canada

[2] LAMA, Université Gustave Eiffel, CNRS, 5, boulevard Descartes Cité Descartes - Champs-sur-Marne, 77454 Marne-la-Vallée cedex 2, France

Published online: 23 April 2021

JEL Classification C18 · C41 · C44 · C54 · C57 · C61 · C63 · C68 · C70 · C90 · D40 · D70 · D83

1 Introduction

1.1 An Historical Overview

Reinforcement learning is related to the study of how agents, animals, autonomous robots use experience to adapt their behavior in order to maximize some rewards. It differs from other types of learning (such as unsupervized or supervised) since the learning follows from feedback and experience (and not from some fixed training sample of data). Thorndike (1911) or Skinner (1938) used reinforcement learning in the context of behavioral psychology, ethology and biology. For instance, Thorndike (1911) studied learning behavior in cats, with some popular experiences, using some 'puzzle box' that can be opened (from the inside) via various mechanisms (with latches and strings) to obtain some food that was outside the box. Edward Thorndike observed that cats usually began experimenting – by pressing levers, pulling cords, pawing, etc. – to escape, and over time, cats will learn how particular actions, repeated in a given order, could lead to the outcome (here some food). To be more specific, it was necessary for cats to explore alternative actions in order to escape the puzzle box. Over time, cats did explore less, and start to exploit experience, and repeat successful actions to escape faster. And the cat needed enough time to explore all techniques, since some could possibly lead more quickly – or with less effort – to the escape. Thorndike (1911) proved that there was a balance between exploration and exploitation. This issue could remind us of the *simulated annealing* in optimization, where a classical optimization routine is pursued, and we allow to move randomly to another point (which would be the exploration part) and start over (the exploitation part). Such a procedure reinforces the chances of converging towards a global optimum, instead of converging to a more local one.

Another issue was that a multi-action sequence was necessary to escape, and therefore, when the cat was able to escape at the first time it was difficult to assign which action actually caused the escape. An action taken at the beginning (such as pulling a string) might have an impact some time later, after other actions are performed. This is usually called a credit assignment problem, as in Minsky (1961). Skinner (1938) refined the puzzle box experiment, and introduced the concept of *operant conditioning* (see Jenkins 1979 or Garcia 1981 for an overview). The idea was to modify a part, such as a lever, such that at some points in time pressing the lever will provide a positive reward (such as food) or a negative one (i.e. a punishment, such as electric shocks). The goal of those experiments was to understand how past voluntary actions modify future ones. Those experiments were performed on rats, and no longer cats. Tolman (1948) used similar experiments (including also mazes) to prove that the classical approach, based on chaining of stimulus-responses, was maybe not the good one to model animal (and men) behaviors. A pure stimulus-responses learning could not be used by rats to escape a maze, when experimenters start to block roads with obstacles. He introduced the idea of *cognitive maps* of the maze that allow for more flexibility. All those techniques could be related to the ones used in reinforcement learning.

Reinforcement learning is about understanding how agents might learn to make optimal decisions through repeated experience, as discussed in Sutton and Barto (1981). More formally, agents (animals, humans or machines) strive to maximize some long-term reward, that is the cumulated discounted sum of future rewards, as in classical economic models. Even if animals can be seen as have a short-term horizon, they do understand that a punishment followed by a large reward can be better than two small rewards, as explained in Rescorla (1979), that introduced the concept of second-order conditioning. A technical assumption, that could be seen as relevant in many human and animal behaviors, is that the dynamics satisfies some Markov property, and in this article we will focus only on Markov decision processes. Reinforcement learning is about solving the credit assignment problem by matching actions, states of the world and rewards.

As we will see in the next section, formally, at time t, the agent at state of the world $s_t \in \mathcal{S}$ makes an action $a_t \in \mathcal{A}$, obtains a reward $r_t \in \mathcal{R}$ and the state of the world becomes $s_{t+1} \in \mathcal{S}$. A policy is a mapping from $\mathcal{S}$ to $\mathcal{A}$, and the goal is to learn from past data (past actions, past rewards) how to find an optimal policy. A popular application of reinforcement learning algorithms is in games, such as playing chess or Go, as discussed in Silver et al. (2018), or Igami (2017) which provides economic interpretation of several algorithms used on games (Deep Blue for chess or AlphaGo for Go) based on structural estimation and machine (reinforcement) learning. More simply, Russell and Norvig (2009) introduced a grid world to explain heuristics about reinforcement learning, see Fig. 1. Positions on the 4×3 grid are the states $\mathcal{S}$, and actions $\mathcal{A}$ are movements allowed. The optimal policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ is here computed using sequential machine learning techniques that we will describe in this article.

1.2 From Machine to Reinforcement Learning

Supervised Machine Learning techniques is a static problem: given a dataset $\mathcal{D}_n = \{(y_i, x_i)\}$, the goal is to learn a mapping $\widehat{m}_n$ between x and y. In decision theory $\widehat{m}_n$ typically takes values in a binary space, which could be to accept or reject a mortgage in credit risk models, or to invest or not in some specific asset. $\widehat{m}_n$ can also take values in the real line, and denote an amount of money to save, a quantity to purchase or a price to ask. Online learning is based on the assumption that (y_i, x_i) arrive in a sequential order, and the focus is on the evolution of $\widehat{m}_n$ as n growth, updating the training dataset from $\mathcal{D}_{n-1}$ to $\mathcal{D}_n$. Reinforcement learning incorporates the idea that at time $n-1$, a choice was made, that will influence (y_n, x_n), and the standard i.i.d. assumption of the dataset is no longer valid. Reinforcement learning is related to sequential decision making and control.

Consider an online shop, where the retailer tries to maximize profit by sequentially suggesting products to consumers. Consumers are characterized by some features, such as their age, or their gender, as well as information about what's

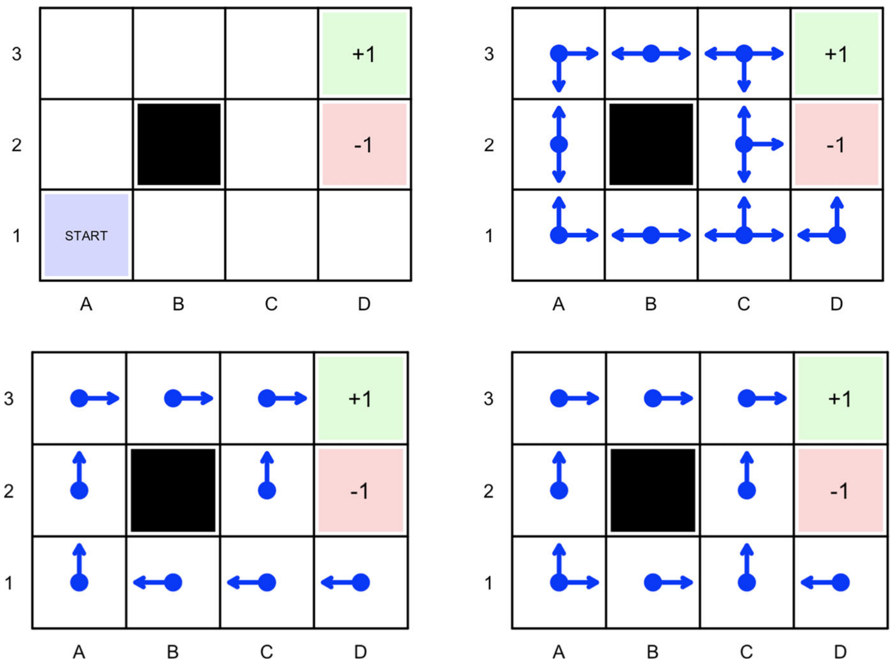

Fig. 1 Sequential decision making problem on a 4×3 grid ($\mathcal{S}$ states), from Russell and Norvig (2009). The agent starts at the state (A,1), and moves around the environment, trying to reach terminal state (D,3) to get a $+1$ reward - and to avoid terminal state (D,2) where a -1 reward (punishment) is given. Possible actions ($\mathcal{A}$) are given on the top-right figure. On the bottom, two policies are given with $\pi : \mathcal{S} \rightarrow \mathcal{A}$ on the left, and $\pi : \mathcal{S} \rightarrow A \subset \mathcal{A}$ on the right. In the later case, there can be random selection of actions in some states, for instance $\pi((\mathsf{A}, 1)) \in \{\text{up}, \text{right}\}$

in their shopping cart. The consumer and the shop will have sequential interactions. Each round, the consumer can either add a product to the shopping cart, or not buy a product and continue shopping, or finally stop shopping and check out. Those transitions are characterized by transition probabilities, function of past states and actions. Such transition probability function is unknown and must be learned by the shop. Should the retailer display the most profitable products, exploiting information he obtained previously, or explore actions, that could be less profitable, but might provide relevant information ?

The induced problems are related to the fact that acting has consequences, possibly delayed. It is about learning to sacrifice small immediate rewards in order to gain larger long-term ones. If standard Machine Learning is about learning from given data, reinforcement learning is about active experimentation. Actions can be seen as an intervention, so there are strong connections between reinforcement learning and causality modeling. Reinforcement learning allows us to infer consequences of interventions (or actions) used in the past. Pearl (2019) asked the simple economic question '*what will happen if we double the price*' (of an item we try to sell)? '*Such questions cannot be answered from sales data alone, because they involve a change in customers behaviour, in reaction to the new pricing*'.

Reinforcement learning is related to such problem: inferring the impact of interventions. And the fact that intervention will impact the environment, mentioned by Pearl (2019), is precisely what reinforcement learning is about. So this theory, central in decision science will appear naturally in sequential experimentation, optimization, decision theory, game theory, auction design, etc. As we will see in the article (and as already mentioned in the previous section), models in sequential decision making as long history in economics, even if rarely mentioned in the computational science literature. Most of the articles published in economic journal mentioned that such problems were computationally difficult to solve. Nevertheless, we will try to show that recent advances are extremely promising, and it is now to possible to model more and more complex economic problems.

1.3 Agenda

In Sect. 2, we will explain connections between *reinforcement learning* and various related topics. We will start with *machine learning* principles, defining standard tools that will be extended later one (with the loss function, the risk of an estimator and regret minimization), in Sect. 2.1. In Sect. 2.2, we introduce dynamical problems with *online learning*, where we exploit past information sequentially. In Sect. 2.3, we present briefly the *multi-armed bandit* problem, where choices are made, at each period of time, and those have consequences on the information we obtain. And finally, in Sect. 2.4 we start formalizing *reinforcement learning* models, and give a general framework. In those sections, we mainly explain the connections between various learning terms used in the literature.

Then, we present various problems tackled in the literature, in Sect. 3. We will start with some general mathematical properties, giving various interpretations of the optimization problem, in Sect. 3.1. Finally, we will conclude, in Sect. 3.4, with a presentation of a classical related problem, called inverse reinforcement learning, where we try to use observed decisions in order to infer various quantities, such as the reward or the policy function.

Finally, three sections are presenting applications of reinforcement learning. In Sect. 4.1, we discuss applications in economic modeling, starting with the classical consumption and income dynamics, which is a classical optimal control problem in economics. We then discuss bounded rationality and strong connections with reinforcement learning. Then we will see, starting from Jovanovic (1982), that reinforcement learning can be used to model single firm dynamics. And finally, we present connections with adaptative design for experiments, inspired by Weber (1992) (and multi-armed bandits).

In Sect. 4.2, we discuss applications of reinforcement learning in operation research, such as the traveling salesman, where the standard dilemma exploration/exploitation can be used to converge faster to (near) optimal solutions. Then we discuss stochastic games and equilibrium, as well as mean-field games, and auctions and real-time bidding. Finally, we will extend the single firm approach of the previous section to the case of oligopoly and dynamic games.

Finally, in Sect. 4.3, we detail applications in finance. We start with risk management, valuation and hedging of financial derivatives problems on then focus on portfolio allocation issues. At last, we present a very natural framework for such algorithms: market impact and market making.

2 From Machine to Reinforcement Learning

Machine learning methods generally make decision based on known properties learned from the training data, using many principles and tools from statistics. Most learning problems could be seen as an optimization of a cost: minimizing a loss or maximizing a reward. However, machine learning models aspire to find generalized predictive pattern, learning algorithms seek to optimize a criterion (loss, reward, regret) on training *and* unseen samples.

2.1 Machine Learning principles

Machine learning has so many branches (supervised vs unsupervised learning, online or not,...) that it is not always easy to identify the label associated to a given real world problem. Therefore, seeing machine learning as a set of data and an optimization criterion is often helpful. To introduce Reinforcement Learning (RL), we propose here a regret approach, which ties machine learning, online aggregation, bandits and, more generally, reinforcement learning.

In order to introduce most of machine learning terminology and schemes, we detail a class of models: supervised learning. In this class of models, one variable is the variable of interest, denoted y and usually called the endogeneous variable in econometrics. To do so, consider some learning sample $\mathcal{D}_n = \{(x_1, y_1), ..., (x_n, y_n)\}$ seen as realizations of n i.i.d. random variables (Y, X). We wish to map the dataset $\mathcal{D}_n$ into a model from the (supposed) statistical relations between x_i and y_i that are relevant to a task. Note that in the context of sequential data we will prefer the generic notation (y_t, x_t).

The goal, when learning, is to find a function $f \in \mathcal{F}$ from the input space $\mathcal{X}$ into the action space $\mathcal{A}$: $f : \mathcal{X} \mapsto \mathcal{A}$. Thus, $f(x|\mathcal{D}_n)$ is the action at some point x. An action could be a prediction (for example what temperature will it be tomorrow? Is there a cat on this image?) or a decision (a chess move, go move...). Note that in a standard regression problem $\mathcal{A}$ is the same as $\mathcal{Y}$, but not necessarily in a classification problem: in a logistic regression, $\mathcal{Y} = \{0, 1\}$ but actions can be probabilities $\mathcal{A} \in [0, 1]$.

The decision function f is all the better as its actions $f(x)$ are good when confronted to the unseen corresponding output y from $\mathcal{Y}$. The loss function (or cost) measures the relevance of these actions when $f(x)$ is taken and y has occurred: $\ell : \mathcal{A} \times \mathcal{Y} \mapsto \mathbb{R}_+$.

The risk is the expectation of the loss:

$$\mathcal{R}(f) = \mathbb{E}\big[\ell(f(X), Y)\big] \tag{1}$$

Thus formalized, the learning is seen as an optimization problem. We wish to find a function $f^* \in \mathcal{F}$ which minimizes the cost:

$$\mathcal{R}(f^*) = \inf_{f \in \mathcal{F}} \{\mathcal{R}(f)\} \tag{2}$$

If such a function f^* exists and is unique it is called oracle or target.

In most applications we do not know the distribution of the data. However, given a training set $\mathcal{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, we use the empirical distribution of the training data and define

$$\widehat{\mathcal{R}}_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i). \tag{3}$$

Thus, we minimize this empirical risk while trying to avoid over-fitting and keeping in mind that the real objective is to minimize $\mathcal{R}(f)$, i.e. the average loss computed on any additional observations. The main difficulty is that the target function is only defined at the training points.

One way to evaluate the learning performance is to compute regret. Regret is defined as the difference between the actual risk, and the optimal oracle risk,

$$\begin{aligned} R &= \mathcal{R}(f) - \mathcal{R}(f^*) \\ &= \mathcal{R}(f) - \inf_{f \in \mathcal{F}} \{\mathcal{R}(f)\} \\ &= \mathbb{E}\big[\ell(f(X), Y)\big] - \mathbb{E}\big[\ell(f^*(X), Y)\big]. \end{aligned}$$

In supervised learning, we prefer the name of excess risk, or excess loss. This notion of regret is particularly relevant in sequential learning, where your action at t depends on previous ones on $t-1, t-2, \ldots$. In online (or sequential) learning, the regret is measured by the cumulative loss it suffers along its run on a sequence of examples. We could see it as the excess loss for not consistently predicting with the optimal model.

$$R_T = \frac{1}{T} \sum_{t=1}^{T} \ell(f_t(x_t), y_t) - \inf_{f \in \mathcal{F}} \left\{ \frac{1}{T} \sum_{t=1}^{T} \ell(f(x_t), y_t) \right\}, \tag{4}$$

where the first term is the estimation error between the target and the prediction, the second is the approximation error. Bandits and Reinforcement Learning deal with maximizing a reward, instead of minimizing a loss. Thus, we can re-write regret as the difference between the reward that could have been achieved and what was actually achieved according to a sequence of actions

$$R_T = \max_a \left\{ \frac{1}{T} \sum_{t=1}^{T} r(a) \right\} - \frac{1}{T} \sum_{t=1}^{T} r(a_t)$$

Thus, minimizing a loss or maximizing a reward is the same optimization problem as minimizing the regret, as defined in Robbins (1952).

For instance, in the ordinary least squares regression, $\mathcal{A} = \mathcal{Y} = \mathbb{R}$, and we use the squared loss: $\ell : (a, y) \mapsto (a - y)^2$. In that case, the mean squared risk is $\mathcal{R}(f) = \mathbb{E}\left[(f(X) - Y)^2\right]$ while the target is $f^*(X) = \mathbb{E}[Y|X]$. In the case of classification, where y is a variable in K categories, $\mathcal{A}$ can be a selection of a class, so $\mathcal{A} = \mathcal{Y} = \{1, \ldots, K\}$. The classical loss in that case is the missclassification dummy loss $\ell(a, y) = 1_{a \neq y}$, and the associated risk is the misspecification probability, $\mathcal{R}(f) = \mathbb{E}\left[1_{f(X) \neq Y}\right] = \mathbb{P}(f(X) \neq Y)$, while the target: is $f^*(X) = \underset{1 \leq k \leq K}{\operatorname{argmax}}\{\mathbb{P}(Y = k|X)\}$.

To go further, Mullainathan and Spiess (2017), Charpentier et al. (2018) or Athey and Imbens (2019) recently discussed connections between econometrics and machine learning, and possible applications of machine learning techniques in econometrics.

2.2 Online learning

In classical (or batch) learning described previously, we want to build an estimator $\widehat{f}$ from $\mathcal{D}_n = \{(x_i, y_i)\}$ such as the regret $\mathbb{E}[\mathcal{R}(\widehat{f})] - \inf_{f \in \mathcal{F}}\{R(f)\}$ is as small as possible. However, in the online learning framework, we get the data through a sequential process and the training set is changing at each iteration. Here, observations are not i.i.d, and not necessarily random.

Following Bottou (1998), assume that data become available at a sequential order, and the goal is to update our previous predictor with the new observation. To emphasize the dynamic procedure, let t denote the number of available observations (instead of n, in order to emphasize the sequential aspect of the problem). Formally, from our samples $\mathcal{D}_t = \{(y_1, x_1), \cdots, (y_t, x_t)\}$ we can derive a model $f(x|\mathcal{D}_t)$, denoted f_t. The goal in online learning is to compute an update f_{t+1} of f_t using the new observation (y_{t+1}, x_{t+1}).

At step t, the learner gets $x_t \in \mathcal{X}$ and predicts $\widehat{y}_t \in \mathcal{Y}$, exploiting past information $\mathcal{D}_{t-1}$. Then, the real observation y_t is revealed and generates a loss $\ell(\widehat{y}_t, y_t)$. Thus, $\widehat{y}_t$ is a function of $\left(x_t, (x_i, y_i)_{i=1\ldots t-1}\right)$.

Consider the case of forecasting with expert advice: expert aggregation. Assume here, as before, a sequential model. We want to predict element by element a sequence of observations $y_1, \ldots, y_T$. At each step t, K experts provide their forecasts $\widehat{y}_{1,t}, \ldots, \widehat{y}_{K,t}$ for the next outcome y_t. Quite naturally, it is possible to build a linear combination (or a weighted average) of those models

$$\widehat{y}_t = \sum_{k=1}^{K} \omega_{k,t} \widehat{y}_{k,t}$$

A natural question is the optimal choice of the weights $\omega_{k,t}$. The aggregation weights expert's prediction $\widehat{y}_{k,t}$ according to a rule in order to build its own forecast $\widehat{y}_t$. The weighting process is online: each instant t, the rule adapts the weights to the past observations and the accuracy of their respective experts, measured by the loss function for each expert $\ell(y_t, \widehat{y}_{k,t})$.

Here, the oracle (or target) is the optimal expert aggregation rule. The prediction $\widehat{y}^*$ use best possible weight combination by minimizing the loss. The empirical regret of the aggregation rule f is defined by:

$$R_T = \frac{1}{T}\sum_{t=1}^{T}\ell(\widehat{y}_t^*, y_t) - \inf_{\omega\in\Omega}\left\{\frac{1}{T}\sum_{t=1}^{T}\ell(\widehat{y}_t, y_t)\right\}$$

where the first term is the estimation error between the target and the prediction, and the second is the approximation error.

There exist several rules for aggregation, the most popular one is probably the Bernstein Online Aggregator (BOA Wintenberger 2017), described in Algorithm 3, which is optimal with bounded iid setting for the mean squared loss.

```
Algorithm 1: Bernstein Online Aggregator (BOA).
  Data: Learning rate $\gamma$
  Result: Sequence $\omega_{k,t} > 0$ for $k = 1..K$ and $t = 0..T$
  Initialization: $\omega_{k,0} \leftarrow$ uniform weights (e.g. $1/K$);
  for $t \in \{1, 2, \dots, n\}$ do
      $\ell_{k,t} \leftarrow \ell(y_t, \widehat{y}_{k,t-1}) - \mathbb{E}_{\pi_{t-1}}[\ell(y_t, \widehat{y}_{k,t-1})]$;
      $\omega_{k,t} \leftarrow \frac{\exp(-\gamma\ell_{k,t}(1+\gamma\ell_{k,t}))\,\omega_{k,t-1}}{\mathbb{E}_{\pi_{t-1}}[\exp(-\gamma\ell_{k,t}(1+\gamma\ell_{k,t}))]}$;
  end
```

This technique, also called *ensemble prediction*, based on aggregation of predictive models, gives an easy way to improve forecasting by using expert forecasts directly. In the context of energy markets, O'Neill et al. (2010) shows that a model based on aggregation of simple ones can reduce residential energy cost and smooths energy usage. Devaine et al. (2013) forecasts electricity consumption using different aggregation rules with specialized experts, and proposes ideas to improve the online mix by varying expert training. Levina et al. (2009) considered the case where a supplier predicts consumer demand by applying an aggregating algorithm to a pool of online predictors.

2.3 Bandits

A related problem is the one where an agent have to choose, repeatedly, among various options but with incomplete information. Multi-armed bandits come from one-armed bandit used in casinos. Imagine an agent playing with several slot machines, each one having a different (unknown) probability of reward associated with. The game is seen as a sequence of single arm pull action and the goal is to maximize its cumulative reward. What could be the optimal strategy to get the highest return?

In order to solve this problem and find the best empirical strategy, the agent has to explore the environment to figure out which arm gives the best reward, but at the same time must choose most of the time the empirical optimal one. It is the exploration-exploitation trade-off: each step either searching for new actions or exploiting the current best one.

The one-armed bandit problem was used in economics in Rothschild (1974), when trying to model the strategy of a single firm facing a market with unknown demand. In an extension, Keller and Rady (1999) consider the problem of the monopolistic firm facing an unknown demand that is subject to random changes over time. Note that the case of several firms experimenting independently in the same market was addressed in McLennan (1984). The choice between various research projects often takes the form of a bandit problem. In Weitzman (1979), each arm represents a distinct research project with a random reward associated with it. The issue is to characterize the optimal sequencing over time in which the projects should be undertaken. It shows that as novel projects provide an option value to the research, the optimal sequence is not necessarily the sequence of decreasing expected rewards. More recently, Bergemann and Hege (1998) and Bergemann and Hege (2005) model venture, or innovation, as a Poisson bandit model with variable learning intensity.

Bandits are a subset of models in online learning and benefits of theoretical results under strong assumptions, most of the time too strong for real-world problems. The multi-armed bandit problem, originally described by Robbins (1952), is a statistical decision model of an agent trying to optimize his decisions while improving his information at the same time. The multi-armed bandit problem and many variations are presented in detail in Gittins (1989) and Berry and Fristedt (1985). An alternative proof of the main theorem, based on dynamic programming can be found in Whittle (1983). The basic idea is to find for every arm a retirement value, and then to choose in every period the arm with the highest retirement value.

In bandits, the information that the learner gets is more restraint than in general online learning: the learner has only access to the cost (loss or reward). At each step t, the learner choose $\widehat{y}_t \in \{1, \ldots, K\}$. Then the loss vector $(\ell_t(1), \ldots, \ell_t(K))$ is established. Eventually, the learner has access to $\ell_t(\widehat{y}_t)$.

Such a problem is called $|\mathcal{A}|$*−multi-armed bandit* in the literature, where $\mathcal{A}$ is the set of action. The learner has K arms, i.e K probability distributions $(v_1, \ldots, v_K)$. Each step t, the agent pulls an arm $a_t \in \{1, \ldots, K\}$ and receives a reward r_t following the probability distribution $v_{a_t=k}$. Let μ_k be the mean reward of distribution v_k, it is the true value when action $a_t = k$ is selected. The estimated value of an action a_t is the expected reward $Q(a_t) = \mathbb{E}[r_t|a_t]$: if action a_t at t is referring to picking the k-th arm of the slot machine, then $Q(a_t) = \mu_k$. The goal is to maximize the cumulative rewards $\sum_{t=1}^{T} r_t$. The bandit algorithm is thus a sequential sampling strategy: $a_{t+1} = f_t(a_t, r_t, \ldots, a_1, r_1)$.

To measure the bandit algorithm performance, we use the previous defined regret. Maximizing the cumulative reward becomes minimizing the potential regret, i.e. the loss of not choosing the optimal actions.

We note $\mu^* = \max_{a \in \{1,\ldots,K\}} \{\mu_a\}$ and the optimal policy is

$$a^* = \operatorname*{argmax}_{a \in \{1,\ldots,K\}} \{\mu_a\} = \operatorname*{argmax}_{a \in \{1,\ldots,K\}} \{Q(a)\}. \tag{5}$$

The regret of a bandit algorithm is thus:

$$R_v(\mathcal{A},T) = T\mu^* - \mathbb{E}\left[\sum_{t=1}^{T} r_t\right] = T\mu^* - \mathbb{E}\left[\sum_{t=1}^{T} Q(a_t)\right] \tag{6}$$

where the first term is the sum of rewards of the oracle strategy which always selects a^*, and the second is the cumulative reward of the agent's strategy.

What could be an optimal strategy ? To get a small regret, a strategy should not select to much sub-optimally arms, i.e. $\mu^* - \mu_a > 0$, which requires to try all arms to estimate the values of these gaps. This leads to the exploration exploitation trade-off previously mentioned. Betting on the current best arm $a_t = \text{argmax}\ \{\mu_{a_t}\}$ is called *exploitation*, while checking that no other arm are better $a_t \neq \text{argmax}\ \{\mu_{a_t}\}$ to find a lower gap is called *exploration*. This process is called a *greedy* action, since it might also be interesting to *explore* by selecting a non-optimal action that might improve our estimation.

For essentially computational reasons (mainly keeping record of all the rewards on the period), it is preferred to write the value function in an incremental expression, as described in Sutton and Barto (1998)

$$Q_{t+1} = \frac{1}{t}\sum_{i=1}^{t} r_i = \frac{1}{t}((t-1)Q_t + r_t) = Q_t + \frac{1}{t}(r_t - Q_t) \tag{7}$$

This leads to the general update rule:

$$\text{New Estimate} = \text{Old Estimate} + \text{Step Size (Target - Old Estimate)}, \tag{8}$$

where Target is a noisy estimate of the true target, and StepSize may depend on t and a. This value function expression, which also identifies to a gradient descent, has already be observed in 2.2 concerning expert aggregation and is mentioned again in the following.

Recently, Misra et al. (2019) consider the case where sellers must decide, on real-time, prices for a large number of item, with incomplete demand information. Using experiments, the seller learns about the demand curve and the profit-maximizing price. The multi-armed bandit algorithms provides an automated pricing policy, using a scalable distribution-free algorithm.

2.4 Reinforcement Learning: A Short Description

In the context of prediction and games (tic-tac-toe, chess, go, or video games), choosing the 'best' move is tricky. Creating data sets for the training process like in the previous approaches (possibly using random simulations) is too costly, since ideally we would like to get all possible actions (positions on the go board or hands of cards). As mentioned in Goodfellow et al. (2016, page 105), "*some machine learning algorithms do not just experience a fixed dataset. For example, reinforcement learning algorithms interact with an environment, so there is a feedback loop between the learning system and its experiences*".

2.4.1 The concepts

In Reinforcement Learning, as in Multi-armed Bandits, data is available at sequential order. However, in the bandits case the action does not impact the agent state, whereas the RL action depends on the environment: an action at a certain state could give a different reward re-visiting the same state. More specifically, at time t

- the learner takes an *action* $a_t \in \mathcal{A}$
- the learner obtains a (short-term) *reward* $r_t \in \mathcal{R}$
- then the *state* of the world becomes $s_{t+1} \in \mathcal{S}$

The states $\mathcal{S}$ refer to the different situations the agent might be in. In the maze example, the location of the rat is a state of the world. The actions $\mathcal{A}$ refer to the set of options available to the agent at some point in time, across all states of the world, and therefore, actions might depend on the state. If the rat is facing a wall, in a dead-end, the only possible action is usually to turn back, while, at some crossroad, the rat can choose various actions. The rewards set $\mathcal{R}$ refer to how rewards (and possibly punishments) are distributed. It can be deterministic, or probabilistic, so in many cases, agents will compute expected values of rewards, conditional on states and actions. These notations were settled in Sutton and Barto (1998), where the goal is to maximize rewards, while previously, Bertsekas and Tsitsiklis (1996) suggested to minimze costs, with some cost-to-go functions.

As in Bandits, the interaction between the environment and the agent involves a trajectory (called also episode). The trajectory is characterized by a sequence of states, actions and rewards. The initial state leads to the first action which gives a reward; then the model is fed by a new state followed by another action and so on.

To determine the dynamics of the environment, and thus the interaction with the agent, the model relies on transition probabilities, based on past states as well as past actions. Nevertheless, with the Markov assumption, we assume that transition probabilities depend only on the current state and action, and not the full history.

Let T be a transition function $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ where:

$$\mathbb{P}\left[s_{t+1} = s' \middle| s_t = s, a_t = a, a_{t-1}, a_{t-2}, \ldots\right] = T(s,a,s'). \tag{9}$$

As a consequence, when selecting an action a, the probability distribution over the next states is the same as the last time we tried this action in the same state.

A *policy* is an action, decided at some state of the world. Formally policies are mapping from $\mathcal{S}$ into $\mathcal{A}$, in the sense that $\pi(s) \in \mathcal{A}$ is an action chosen in state $s \in \mathcal{S}$. Note that stochastic policies can be considered, and in that case, π is a $\mathcal{S} \times \mathcal{A} \to [0,1]$ function, such that $\pi(a,s)$ is interpreted as the probability to chose action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. The set of policies is denoted Π.

After time step t, the agent receives a reward r_t. The goal is to maximize its cumulative reward in the long run, thus to maximize the expected return. Resuming (Sutton and Barto 1998), we can defined the return as the sum of the reward:

$$G_t = \sum_{k=t+1}^{T} r_k \tag{10}$$

Unlike Bandits approaches, here the cumulative reward is computed starting from t. Sometimes the agents can receive running reward, associated to tasks where there is no notion of final time step, so we introduce the discounted return:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \tag{11}$$

where $0 \leq \gamma \leq 1$ is the discount factor which gives more importance to recent reward (and can allow G_t to exist). We can also re-write G_t in a recursive (or incremental way too) since $G_t = r_{t+1} + \gamma G_{t+1}$.

To quantify the performance of an action, we introduce, as in the previous section, the action-function, or Q-value on $\mathcal{S} \times \mathcal{A}$:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\mathbb{P}}\left[G_t \middle| s_t, a_t, \pi\right] \tag{12}$$

In order to maximize the reward, the optimal strategy is characterized by the optimal policies

$$\pi^{\star}(s_t) = \underset{a\in\mathcal{A}}{\text{argmax}}\left\{Q^{\star}(s_t, a)\right\}. \tag{13}$$

That function can be used to derive an optimal policy, and the optimal value function producing the best possible return (in sense of regret):

$$Q^{\star}(s_t, a_t) = \max_{\pi\in\Pi}\left\{Q^{\pi}(s_t, a_t)\right\}. \tag{14}$$

Considering optimal strategy and regret leads to the previously mentioned exploration exploitation trade-off. As seen in the bandits Sect. 2.3, the learner try various actions to explore the unknown environment in order to learn the transition function T and the reward R. The exploration is commonly implemented by ε-greedy algorithm (described in the bandits Sect. 2.3), as in Monte-Carlo methods or Q-learning.

Bergemann and Välimäki (1996) provided a nice economic application of the exploration-exploitation dilemma. In this model, the true value of each seller's product to the buyer is initially unknown, but additional information can be gained by experimentation. When assuming that prices are given exogeneously, the buyer's problem is a standard multi-armed bandit problem. The paper in nevertheless original since the cost of experimentation is here endogenized.

2.4.2 An Inventory Illustration

A classical application of such framework is the control of inventory, with limited size, when the demand is uncertain. Action $a_t \in \mathcal{A}$ denote the number of ordered items arriving on the morning of day t. The cost is $\underline{p}a_t$ if the individual price of items is $\underline{p}$ (but some fixed costs to order items can also be considered). Here $\mathcal{A} =$

$\{0, 1, 2, \ldots, m\}$ where m is the maximum size of storage. States $s_t = \mathcal{S}$ are the number of items available at the end of the day (before ordering new items for the next day). Here also, $\mathcal{S} = \{0, 1, 2, \ldots, m\}$. Then, the state dynamics are

$$s_{t+1} = \big(\min\{(s_t + a_t), m\} - \varepsilon_t\big)_+$$

where ε_t is the unpredictable demand, independent and identically distributed variables, taking values in $\mathcal{S}$. Clearly, (s_t) is a Markov chain, that can be described by its transition function T,

$$T(s, a, s') = \mathbb{P}\big[s_{t+1} = s' \big| s_t = s, a_t = a\big] = \mathbb{P}\big[\varepsilon_t = \big(\min\{(s + a), m\} - s'\big)_+\big]$$

The reward function R is such that, on day t, revenue made is

$$r_t = -\underline{p}a_t + \overline{p}\varepsilon_t = -\underline{p}a_t + \overline{p}\big(\min\{(s_t + a_t), m\} - s_{t+1}\big)_+ = R(s_t, a_t, s_{t+1})$$

where $\overline{p}$ is the price when items are sold to consumers (and $\underline{p}$ is the price when items are purchased). Note that in order to have a more interesting (and realistic) model, we should introduce fixed costs to order items, as costs to store item. In that case

$$r_t = -\underline{p}a_t + \overline{p}\big(\min\{(s_t + a_t), m\} - s_{t+1}\big)_+ - k_1 \mathbf{1}_{a_t > 0} - k_2 s_t,$$

for some costs k_1 and k_2. Thus, reinforcement learning will appear quite naturally in economic problems, and as we will see in the next section, several algorithms can be used to solve such problems, especially when some quantities are unknown, and can only be estimated... assuming that enough observations can be collected to do so.

3 Reinforcement Learning

Now that most of essential notions have been defined and explained, we can focus on Reinforcement Learning principles, and possible extensions. This section deals with the most common approaches, their links with ordinary economy or finance problems and, eventually, some known difficulties or constraints.

3.1 Mathematical Context

Classically, a Markov property is assumed on the reward and the observations. A Markov decision process (MDP) is a collection $(\mathcal{S}, \mathcal{A}, T, r, \gamma)$ where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, T the transition function $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, R is a reward function $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ and $\gamma \in [0, 1)$ is some discount factor. A policy $\pi \in \Pi$ is a mapping from $\mathcal{S}$ to $\mathcal{A}$. Each time t, the agent takes one of many *actions* $a_t \in \mathcal{A}$, obtains a (short-term) *reward* $r_t \in \mathcal{R}$, and then switches from *state* s_t to $s_{t+1} \in \mathcal{S}$.

```
Algorithm 2: Policy generation
    Data: transition function $T$ and policy $\pi$
    Result: Sequence $(a_t, s_t)$
    initialization: $s_1 \leftarrow$ initial state;
    for $t \in \{1, 2, \dots\}$ do
        $a_t \leftarrow \pi(s_t) \in \mathcal{A}$ ;
        $s_{t+1} \leftarrow T(s_t, a_t, \cdot) = \mathbb{P}\big[s_{t+1} = \cdot \big| s_t, a_t, \big] \in \mathcal{S}$ ;
    end
```

The expected reward given a policy π and starting from state $s \in \mathcal{S}$ at time t, is

$$V^{\pi}(s_t) = \mathbb{E}_{\mathbb{P}}\left(\sum_{k \in \mathbb{N}} \gamma^k r_{t+k} \middle| s_t, \pi\right) \tag{15}$$

called *value of a state s under policy* π, where $r_t = \mathbb{E}_a[R(s_t, a, s_{t+1})]$ when $a \sim \pi(s_t, \cdot)$ and $\mathbb{P}$ is such that $\mathbb{P}(S_{t+1} = s_{t+1} | s_t, a_t) = T(s_t, a, s_{t+1})$. Algorithm 2 describes how policy is built sequentially. Since the goal is to find a best policy – that is the policy that receives the most reward – we define

$$V^{\star}(s_t) = \max_{\pi \in \Pi} \left\{ V^{\pi}(s_t) \right\} \tag{16}$$

which is the maximal reward that could be expected in the given set of policies Π. The algorithm 3 indicates a way to evaluate and quantify the relevance of a policy.

```
Algorithm 3: Policy valuation
    Data: policy $\pi$, threshold $\varepsilon > 0$, reward $R(s, a, s')$, $\forall s, a, s'$
    Result: Value of policy $\pi$, $V^{\pi}$
    initialization: $V(s)$ for all $s \in \mathcal{S}$ and $\Delta = 2\varepsilon$;
    while $\Delta > \varepsilon$ do
        $\Delta \leftarrow 0$ for $s \in \mathcal{S}$ do
            $v \leftarrow V(s)$ ;
            $V(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a, s) \sum_{s' \in \int} T(s, a, s') \big[R(s, a, s') + \gamma V(s')\big]$ ;
            $\Delta \leftarrow \max\{\Delta, |v - V(s)|\}$
        end
    end
```

As in Watkins and Dayan (1992), one can define the Q-value (or action-value function) for policy π on $\mathcal{S} \times \mathcal{A}$ as

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\mathbb{P}}\left(\sum_{k \in \mathbb{N}} \gamma^k r_{t+k} \middle| s_t, a_t, \pi\right) \tag{17}$$

which can be written, from Bellman's equation (see Bellman 1957)

$$Q^{\pi}(s_t, a_t) = \sum_{s' \in \mathcal{S}} \left[r(s_t, a_t, s') + \gamma Q^{\pi}(s', \pi(s')) \right] T(s_t, a_t, s') \tag{18}$$

and as previously, let

$$Q^{\star}(s_t, a_t) = \max_{\pi \in \Pi} \left\{ Q^{\pi}(s_t, a_t) \right\}. \tag{19}$$

Observe that $Q^{\pi}(s_t, a_t)$ identifies to the value function in state s_t when playing action a_t at time t and then acting optimally. Hence, knowing the Q-function directly provides the derivation of an optimal policy

$$\pi^{\star}(s_t) = \underset{a \in \mathcal{A}}{\text{argmax}} \left\{ Q^{\star}(s_t, a) \right\}. \tag{20}$$

This optimal policy $\pi^{\star}$ assigns to each states s the highest-valued action. In most applications, solving a problem boils down to computing the optimal policy $\pi^{\star}$.

Note that with finite size spaces $\mathcal{S}$ and $\mathcal{A}$, we can use a vector form for $Q^{\pi}(s, a)$'s, $\boldsymbol{Q}^{\pi}$, which is a vector of size $|\mathcal{S}||\mathcal{A}|$. In that case, Equation (18) can be written

$$\boldsymbol{Q}^{\pi} = \boldsymbol{R} + \gamma \boldsymbol{P\Pi} \boldsymbol{Q}^{\pi} \tag{21}$$

where $\boldsymbol{R}$ is such that

$$\boldsymbol{R}_{(s,a)} = \sum_{s' \in \mathcal{S}} r(s_t, a_t, s') T(s_t, a_t, s')$$

and $\boldsymbol{P\Pi}$ is the matrix of size $|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}||\mathcal{A}|$ that constraints transition probabilities, from (s, a) to $(s', \pi(s'))$ (and therefore depends on policy π).

If we use notations introduced in Sect. 2.4, we have to estimate $Q(s, a)$ for all states s and actions a, or function $V(s)$. Bellman equation on Q^{π} means that V^{π} satisfies

$$V^{\pi}(s_t) = \sum_{s' \in \mathcal{S}} \left[r(s_t, \pi(s_t), s') + \gamma V^{\pi}(s') \right] T(s_t, \pi(s_t), s'). \tag{22}$$

Unfortunately, in many applications, agents have no prior knowledge of reward function r, or transition function T (but do know that it satisfies the Markov property). Thus, the agent will have to *explore* – or perform actions – that will give some feedback, that can be used, or *exploited*.

As discussed previously, Q function is updated using

$$Q(s, a) \leftarrow (1 - \alpha) Q(s, a) + \alpha \big(r(s, a, s') + \gamma \max_{a' \in \mathcal{A}} \left\{ Q(s', a') \right\} \big). \tag{23}$$

A standard procedure for exploration is the ε-greedy policy, mentioned already in the bandit context, where the learner makes the best action with probability $1 - \varepsilon$, and consider a randomly selected action with probability ε. Alternatively, consider some exploration function that will give preference to less-visited states, using some sort of penalty

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r(s,a,s') + \gamma \max_{a'\in\mathcal{A}}\left\{Q(s',a') + \frac{\kappa}{n_{s,a}}\right\}\right). \tag{24}$$

where $n_{s,a}$ denotes the number of times where state (s, a) has been visited, where κ will be related to some exploration rate. Finally, with the Boltzmann exploration strategy, probabilities are weighted with their relative Q-values, with

$$p(a) = \frac{e^{\beta Q(s,a)}}{e^{\beta Q(s,a_1)} + \ldots + e^{\beta Q(s,a_n)}}, \tag{25}$$

for some $\beta > 0$ parameter. On the one hand, with a low value for β, the selection strategy tends to be purely random. On the other hand, with a high value for β, the algorithm selects the action with the highest Q-value, and thus, ceases the experiment.

3.2 Some Dynamical Programming Principles

In Dynamic Programming, as well as in most of Reinforcement Learning problems, we use value functions to choose actions and build an optimal policy. Many algorithms of this field compute optimal policies in a fully known model in a Markov decision process environment. It is not always possible in real-world problems or too computationally expensive. However, Reinforcement Learning lies on several principles of Dynamic Programming and we present here a way to obtain an optimal policy once we have found the optimal value function which satisfy the Bellman equation: the Policy iteration.

3.2.1 Policy Iteration

Value function V^π satifies Equation (22), or to be more specific a system of $|\mathcal{S}|$ linear equations, that can be solved when all functions – T and r – are known. An alternative is to use an iterative procedure, where Bellman's Equation is seen as a updating rule, where V^π_{k+1} is an updated version of V^π_k

$$V^\pi_{k+1}(s_t) = \sum_{s'\in\mathcal{S}} \left[r(s_t,\pi(s_t),s') + \gamma V^\pi_k(s')\right] T(s_t,\pi(s_t),s'). \tag{26}$$

The value function V^π is a fixed point of this recursive equation.

Once we can evaluate a policy π, Howard (1960) suggested a simple iterative procedure to find the optimal policy, called *policy iteration*. The value of action a is obtained using

$$Q^\pi(s_t,a) = \sum_{s'\in\mathcal{S}} \left[r(s_t,a,s') + \gamma V^\pi(s')\right] T(s_t,a,s'), \tag{27}$$

so if $Q^\pi(s_t,a)$ is larger than $V^\pi(s_t)$ for some $a \in \mathcal{A}$, choosing a instead of $\pi(s_t)$ would have a higher value. It is then possible to improve the policy by selecting that better action. Hence, a greedy policy π' can be considered, simply by choosing the best action

$$\pi'(s_t) = \underset{a\in\mathcal{A}}{\operatorname{argmax}}\{Q^\pi(s_t,a)\}. \tag{28}$$

The algorithm suggested by Howard (1960) starts from a policy π_0. Then, at step k, given a policy π_k, the algorithm computes its value V^{π_k}, improves it with π_{k+1}, and eventually iterates.

Unfortunately, such a procedure can be very long, as discussed in Bertsekas and Tsitsiklis (1996). And it assumes that all information is available, which is not the case in many applications. As we will see in the next sections, it is then necessary to sample to learn the model – the transition rate and the reward function.

3.2.2 Policy Iteration Using Least Squares

$Q^\pi(s,a)$ is essentially an unknown function, since it is the expected value of the cumulated sum of discounted future random rewards. As discussed in Sect. 2.2, a natural stategy is to use a parametric model, $Q^\pi(s,a,\boldsymbol{\beta})$ that will approximate $Q^\pi(s,a)$. Linear predictors are obtained using a linear combination of some basis functions,

$$Q^\pi(s,a,\boldsymbol{\beta}) = \sum_{j=1}^{k}\psi_j(s,a)\beta_j = \boldsymbol{\psi}(s,a)^\top\beta_j,$$

for some simple functions ψ_j, such as polynomial transformations. With the notation of Sect. 2.4.1, write $\boldsymbol{Q}^\pi = \boldsymbol{\Psi\beta}$. Thus, substituting in equation (21), we obtain

$$\boldsymbol{\Psi\beta} \approx \boldsymbol{R} + \gamma\boldsymbol{P\Pi\Psi\beta} \text{ or } \big(\boldsymbol{\Phi} - \gamma\boldsymbol{P\Pi\Psi}\big)\boldsymbol{\beta} \approx \boldsymbol{R}.$$

As in Sect. 2.4.1, we have an over-constrained system of linear equations, and the least-square solution is

$$\boldsymbol{\beta}_\star = \big((\boldsymbol{\Psi} - \gamma\boldsymbol{P\Pi\Psi})^\top(\boldsymbol{\Psi} - \gamma\boldsymbol{P\Pi\Psi})\big)^{-1}(\boldsymbol{\Psi} - \gamma\boldsymbol{P\Pi\Psi})^\top\boldsymbol{R}.$$

This is also called *Bellman residual minimizing approximation*. And as proved in Nedić and Bertsekas (2003) and Lagoudakis and Parr (2003), for any policy π, the later can be written

$$\boldsymbol{\beta}_\star = \big(\underbrace{\boldsymbol{\Psi}^\top(\boldsymbol{\Psi} - \gamma\boldsymbol{P\Pi\Psi})}_{=\boldsymbol{A}}\big)^{-1}\underbrace{\boldsymbol{\Psi}^\top\boldsymbol{R}}_{=\boldsymbol{b}}.$$

Unfortunately, when rewards and transition probabilities are not given, we cannot use (directly) the equations obtained above. But some approximation, based on previous t observed values can be used. More precisely, at time t we have a sample $\mathcal{D}_t = (s_i, a_i, r_i)$, and we can use algorithm 4.

```
Algorithm 4: Least square policy iteration
  Data: Policy $\pi$, $\gamma$, sample $\mathcal{D}_t$ and basis functions $\psi_j$
  Result: Optimal $\pi$
  initialization $\widehat{\boldsymbol{A}} \leftarrow \boldsymbol{0}$ and $\widehat{\boldsymbol{B}} \leftarrow \boldsymbol{0}$;
  for $i \in \{1, 2, \cdots, t-1\}$ do
      $\widehat{\boldsymbol{A}} \leftarrow \widehat{\boldsymbol{A}} + \psi(s_i, a_i)\big(\psi(s_i, a_i) - \gamma\psi(s_{i+1}, \pi(s_{i+1}))\big)^\top$ ;
      $\widehat{\boldsymbol{b}} \leftarrow \widehat{\boldsymbol{b}} + \psi(s_i, a_i) r_i$ ;
  end
  $\widehat{\boldsymbol{\beta}}_\star \leftarrow \widehat{\boldsymbol{A}}^{-1}\widehat{\boldsymbol{b}}$ ;
  $\pi^\star(s) \leftarrow \underset{a \in \mathcal{A}}{\text{argmax}} \left\{ \psi(s, a)^\top \widehat{\boldsymbol{\beta}}_\star \right\}$
```

If states and actions are uniformely observed on those t past values, $\widehat{\boldsymbol{A}}$ and $\widehat{\boldsymbol{b}}$ converge respectively towards $\boldsymbol{A}$ and $\boldsymbol{b}$ and therefore, $\widehat{\boldsymbol{\beta}}_\bigstar = \widehat{\boldsymbol{A}}^{-1}\widehat{\boldsymbol{b}}$ is a consistent approximation of $\boldsymbol{\beta}_\bigstar$.

3.2.3 Model-Based versus Model-Free Learning

Model-based strategies are based on a fully known environment. We can learn about the state transition $T(s_t, a_t, s_{t+1}) = \mathbb{P}(S_{t+1} = s_{t+1} | s_t, a_t)$ and the reward function $R(s_t)$ in order to find the optimal solution using Dynamic programming. Starting from s_0, the agent will chose randomly actions in $\mathcal{A}$ at each step. Let (s_i, a_i, s_{i+1}) denote the simulated set of present state, present action and future state. After n generations, the empirical transition is

$$\widehat{T}_n(s, a, s') = \frac{\sum_i \boldsymbol{1}_{(s,a,s')}(s_i, a_i, s_{i+1})}{\sum_i \boldsymbol{1}_{(s,a)}(s_i, a_i)}$$

and

$$\widehat{R}_n(s, a, s') = \frac{\sum_i R(s_i, a_i, s_{i+1})}{\sum_i \boldsymbol{1}_{(s,a,s')}(s_i, a_i, s_{i+1})}$$

By the law of large numbers, $\widehat{T}_n$ and $\widehat{R}_n$ will respectively converge towards T and R, as n goes to infinity. This is the *exploration* part.

That strategy is opposed to so-called *model-free* approaches, where the learning process could deal with incomplete information or the model is just unknown. In the next sections, we describe classical model-free algorithms: Temporal-Difference (TD), Policy Gradient and Actor-Critic.

3.3 Some Solution Methods

Here is presented briefly some common model-free approaches in Reinforcement Learning where the model is unknown or has to learn with incomplete information. For the first one, we focus on one significant breakthroughs in reinforcement learning, the Q-learning (introduced in Watkins (1989)), an off-policy Temporal-Difference (TD) control model in discrete action space. Then, we quickly describe Policy Gradient and Actor-Critic, two methods considering continuous action space.

3.3.1 *Q*-learning

As TD approaches, Q-learning needs to interact with the environment, meaning that it is necessary to simulate the policy and generate samples. Recent works using neural network, like Deep Q-Network (DQN) show impressive results in complex environments, not without training constraints.

Q-learning was introduced in Watkins and Dayan (1992). Bellman Equation (18) is expressed as

$$Q^{\pi}(s_t, a_t) = \sum_{s' \in \mathcal{S}} \left[R(s_t, a_t, s') + \gamma Q^{\pi}(s', \pi(s')) \right] T(s_t, a_t, s'), \tag{29}$$

and the optimal value satisfies

$$Q^{\star}(s_t, a_t) = \sum_{s' \in \mathcal{S}} \left[R(s_t, a_t, s') + \gamma V^{\star}(s') \right] T(s_t, a_t, s') \tag{30}$$

where $V^{\star}(s') = \max_{a' \in \mathcal{A}} \left\{ Q^{\star}(s', a') \right\}$.

Thus, Q-learning is based on the following algorithm: starting from $Q_0(s, a)$, at step $k + 1$ set

$$Q_{k+1}(s, a) = \sum_{s' \in \mathcal{S}} \left[R(s, a, s') + \gamma \max_{a' \in \mathcal{A}} \left\{ Q_k(s', a') \right\} \right] T(s, a, s'). \tag{31}$$

This approach is used in Hasselt (2010) where the Q-function, i.e. value-function, is approximated by a neural network for example.

However, Q-learning has some pitfalls depending of the use-case considered. First, the action space has to be discrete and finite, a strong assumption if we handle prices or risk hedging. Moreover, because the algorithm seeks to figure out the quality of each possible action, if the action space is too large the algorithm might converge to local maxima, or the exploration might be too long. To overcome these difficulties, an idea would be learn the policy directly. Policy Gradient approaches, detailed in the following, simply evaluate the probability of selecting each action and chose the preferred one.

3.3.2 Policy Optimization

In order to avoid computing and comparing the expected return of different actions, as in Q-learning, an agent could learn directly a mapping from states to actions. Here, we try to infer a parameterized policy $\pi(a|s, \theta)$ that maximizes the outcomes reward from an action on an environment. Policy learning converges faster than value-based learning process and allows continuous action space of the agent as the policy is now a parameterized function depending on θ. An infinite number of actions would be computationally too expensive to optimize otherwise.

Algorithm 5: Direct policy search

```
Data: A threshold $\varepsilon$, reward $R(s,a,s')$, $\forall s,a,s'$
Result: Optimal policy $\pi^\star$
initialization: $V(s)$ for all $s \in \mathcal{S}$ and $\Delta = 2\varepsilon$;
while $\Delta > \varepsilon$ do
    $\Delta \leftarrow 0$ for $s \in \mathcal{S}$ do
        $v \leftarrow V(s)$ ;
        $V(s) \leftarrow \max_{a \in \mathcal{A}} \left\{ \sum_{s' \in \mathcal{S}} T(s,a,s') \big[ R(s,a,s') + \gamma V(s') \big] \right\}$ ;
        $\Delta \leftarrow \max\{\Delta, |v - V(s)|\}$;
    end
end
for $s \in \mathcal{S}$ do
    $\pi(s) \leftarrow \operatorname{argmax}_{a \in \mathcal{A}} \left\{ \sum_{s' \in \mathcal{S}} T(s,a,s') \big[ R(s) + \gamma V(s') \big] \right\}$;
end
```

This approach is based on the Policy Gradient Theorem from Sutton and Barto (1998). Algorithm 5 details the Direct Policy Search, where action-function is not computed (as in Q-learning) but only value-function and policy. Some application examples can be found in the application Sect. 4.

3.3.3 Actor-Critic

Actor-Critic aims to take advantage of both Value and Policy approaches. By merging them, it can benefit of continuous and stochastic environments and faster convergence from Policy learning, as well as sample efficiency and steady from Value one. In the Actor-Critic setup, two models interact in order to give the best cumulative reward. Using simultaneously an actor, which updates the policy parameter θ in $\pi(a|s;\theta)$, and a critic which updates the value function $V(s)$ or action-value function $Q(a|s)$, this model is able to learn complex environments as well as complex Value-functions. Tamar et al. (2015) propose risk-sensitive policy gradient methods using a Actor-Critic model, handling popular financial tools such as the variance or conditional value at risk (CVaR).

3.4 Inverse Reinforcement Learning

In the econometric literature, this problem can be found in many articles published in the 80's, such as Miller (1984) in the context of job matching and occupational choice, Pakes and Schankerman (1984) on the rate of obsolescence of patents, and research gestation lags, Wolpin (1984) on the estimation of a dynamic stochastic model of fertility and child mortality, Pakes (1986) on optimal investment strategies or Rust (1987) on replacement of bus engines, where structural models are used to better understand human decision making. Hotz and Miller (1993), Aguirregabiria and Mira (2002) or more recently Magnac and Thesmar (2002) or Su and Judd (2012) mentioned the computational complexity of such algorithms on economic applications.

Most of those approaches are related to the literature on dynamic discrete choice model (see Aguirregabiria and Mira 2010 for a survey, or Semenova 2018 for connections with machine learning tools). In those models, there is a finite set of possible actions $\mathcal{A}$, as assumed also in the previous descriptions, and they focus on conditional choice probability, which is the probability that choosing $a \in \mathcal{A}$ is optimal in state $s \in \mathcal{S}$,

$$\text{ccp}(a|s) = \mathbb{P}[a \text{ is optimal in state } s] = \mathbb{P}\left[\{Q(a,s) \geq Q(a',s), \forall a' \in \mathcal{A}\}\right].$$

Assuming that rewards have a Gumbel distribution, we obtain a multinomial logit model, where the log-odds ratios are proportional to the value function. For instance in the bus-repair problem of Rust (1987), the state s is the mileage of the bus, and the action a is in the set $\{\text{opr}, \text{rep}\}$ (either operate, or replace). Per period, the utility is

$$U_\theta(s_t, \varepsilon_t, a) = \varepsilon_t + u_\theta = \varepsilon_t + \begin{cases} -OC_\theta(s_t) & \text{if } a = \text{opr} \\ -RC - OC_\theta(0) & \text{if } a = \text{rep} \end{cases}$$

where RC is some (fixed) replacing cost, OC_θ is the operating cost (that might depend on some parameter θ), and ε_t is supposed to have a Gumbel distribution. The respective costs are supposed to be known.

Then

$$\text{ccp}_\theta(a|s) = \frac{\exp[v_\theta(s,a)]}{\exp[v_\theta(s,\text{opr})] + \exp[v_\theta(s,\text{rep})]}$$

where $v_\theta(s,a) = u_\theta(s,a) + \beta EV_\theta(s,a)$ where $ES_\theta(s,a)$ is the unique solution of

$$EV_\theta(s,a) = \int \log\left[u_\theta(s,\text{opr}) + u_\theta(s',\text{opr}) + \beta(EV_\theta(s,\text{opr}) + EV_\theta(s',\text{rep}))\right] T(s'|s,a)$$

Hotz and Miller (1993) proved that the mapping between conditional choice probabilities and choice specific value function is invertible. As discussed in Su and Judd (2012), based on observed decisions made by the superintendent of maintenance of the bus company, structural estimation is computationally complex.

The main idea of inverse reinforcement learning (or learning from demonstration, as defined in Schaal (1996)) is to learn the reward function based on the agent's decisions, and then find the optimal policy (the one that maximizes this reward function) using reinforcement learning techniques. Similar techniques are related to this idea. In imitation learning (also called behavioral cloning in Bain and Sammut (1995)), we learn the policy using supervised learning algorithms, based on the sample of observations $\{(s_i, a_i)\}$, that is unfortunately not distributed independently and identically in the state-action space. In apprenticeship learning, we try to find a policy that performs as well as the expert policy, as introduced in Abbeel and Ng (2004). Rothkopf and Dimitrakakis (2011) mentioned applications of reinforcement learning on preference elicitation, extended in Klein et al. (2012). See (Ng et al. 2000) for a survey of various algorithms used in inverse reinforcement learning, as well as (Abbeel and Ng 2004).

4 Applications

4.1 Applications in Economic Modeling

If it is possible to find a framework very similar to the one use in reinforcement learning in old economic literature (see for instance the seminal thesis (Hellwig 1973)), as mentioned in Arthur (1991) or Barto and Singh (1991), two survey of reinforcement learning techniques in computational economics, published thirty years ago. Recently, Hughes (2014) updated the survey on applications of reinforcement learning to economic problems with up-to-date algorithms.

4.1.1 Consumption and Income Dynamics

Consider an infinitely living agent, with utility $u(c_t)$ when consuming $c_t \geq 0$ in period t. That agent receives random income y_t at time t, and assume that (y_t) is a Markov process with transition $T(s, s') = \mathbb{P}[y_{t+1} = s' | y_t = s]$. Let w_t denote the wealth of the agent, at time t, so that $w_{t+1} = w_t + y_t - c_t$. Assume that the wealth must be non-negative, so $c_t \leq w_t + y_t$. And for convenience, $w_0 = 0$, as in Lettau and Uhlig (1999). At time t, given state $s_t = (w_t, y_t)$, we seek $c_t^\star$ solution of

$$v(w_t, y_t) = \max_{c \in [0, w_t + y_t]} \left\{ u(c) + \gamma \sum_{y'} \left[v(w_t + y_t - c, y') \right] T(y_t, y') \right\}$$

This is a standard recursive model, discussed in Ljungqvist and Sargent (2018) or Hansen and Sargent (2013), assuming that utility function u is continuous, concave, strictly increasing and bounded, the value function v is itself continuous, concave, strictly increasing and bounded in wealth w_t, and gives a unique decision function $c^\star(w_t, y_t)$. Stokey et al. (1989) extented that model to derive a *general dynamic decision problem* where income y is now a state $s \in \mathcal{S} = \{s_1, \ldots, s_n\}$, and consumption c is now an action $a \in \mathcal{A} = \{a_1, \ldots, a_m\}$. Utility is now a function of (s, a), and it is assume that the state process (s_t) is a Markov chain, with transition matrix $\boldsymbol{T}_a$ (and transition function T_a). The decision problem is written as a dynamic problem

$$v(s) = \max_{a \in \mathcal{A}} \left\{ u(s, a) + \gamma \mathbb{E}_{s' \sim T_a} \left[v(s') \right] \right\}$$

Using contraction mapping theorems, there is a unique solution $v^\star$ to this problem, that can be characterized by some decision function $\pi^\star : \mathcal{S} \mapsto \mathcal{A}$ that prescribes the best action $\pi^\star(s)$ in each state s.

$$v^{\pi}(s) = u(s,\pi(s)) + \gamma \mathbb{E}_{s'\sim T_a}\left[v^{\pi}(s')\right]$$

The solution can be obtained easily using some matrix formulation, $\boldsymbol{v}^{\pi} = (\mathbb{I}_n - \gamma \boldsymbol{T}^{\pi})^{-1}\boldsymbol{u}^{\pi}$, where $\boldsymbol{v}^{\pi} = (v^{\pi}(s_i)) \in \mathbb{R}^n$, $\boldsymbol{T}^{\pi} = [T^{\pi(s_i)}(s_j)]$ is a $n \times n$ matrix, and $\boldsymbol{u}^{\pi} = (s_i, \pi(s_i)) \in \mathbb{R}^n$. Once v^{π} is obtained for any policy π, then $v^{\star}$ is the maximum value. Stokey et al. (1989) gives several rules of thumb to solve that problem more efficiently, inspired by Holland (1986).

In the context of multiple agents, Kiyotaki and Wright (1989) describes an economy with three indivisible goods, that could be stored, but with a cost, and three types of agents, infinitely living, favoring one of the good. In Basci (1999), agents do not know the equilibrium strategies and act according to some randomly held beliefs regarding the values of the possible actions. Agents have opportunities of both learning by experience, and by imitation. Basci (1999) observes that the presence of imitation either speeds up social convergence to the theoretical Markov-Nash equilibrium or leads every agent of the same type to the same mode of suboptimal behavior. We will discuss Nash equilibrium with multiple agents in the next section.

4.1.2 Bounded Rationality

Simon (1972) discussed the limits of the rationality concept, central in most economic models, introducing the notion of bounded rationality, related to various concepts that were studied afterwards, such as bounded optimality (as in Russell and Subramanian (1995) with possible limited thinking time, or memory constraints) or computational rationality (as defined in Gershman et al. (2015)) minimal rationality (such as Cherniak (1986) where minimal sets of conditions to have rationality are studied), ecological or environmental rationality (with a close look at the environment, that will influence decisions, as discussed in Gigerenzer and Goldstein (1996)). More recently, Kahneman (2011) popularized this concept with the two modes of thought: *system 1* is fast, instinctive and emotional while *System 2* is slower, more deliberative, and more logical. Simon (1972) suggests that bounded rationality can be related to uncertainty, incomplete information, and possible deviations from the original goal, emphasizing the importance of heuristics to solve complex problems, also called practical rationality (see Rubinstein 1998 of Aumann 1997 for some detailed survey). Recently, Leimar and McNamara (2019) suggested that adaptive and reinforcement learning leads to bounded rationality, while Abel 2019 motivates reinforcement learning as a suitable formalism for studying boundedly rational agents, since "*at a high level, Reinforcement Learning unifies learning and decision making into a single, general framework*".

Simon (1972) introduce dthe problem of *infinite regress*, where agents are spending more resources on finding the optimal simplification of the problem than solving the original problem. This simplification problem is related to the sparsity issue in standard supervised learning. Gabaix (2014) discussed algorithms for finding a sparse model, either with short range memory, or focusing on *local thinking*, as defined in Gennaioli and Shleifer (2010) (where agents combine data received from the external world with information retrieved from memory to evaluate a hypothesis). Reinforcement learning provides powerful tools to solve complex problems, where agents are suppose to have bounded rationality. And the literature (in reinforcement learning) has developed several measures for evaluating the capacity of an agent to effectively explore its environment. The first one is the regret of an agent, which measures how much worse the agent is relative to the optimal strategy (that could be related to unbounded rationality). The second one is the sample complexity (or computational complexity) which measures the number of samples an agent need before it can act near-optimally, with high probability. In connection with models with bounded rational heterogeneous agents, Granato et al. (2008) investigates the equilibrium properties under adaptive learning. They prove that the conditions for at least one learnable equilibrium are similar to those under homogeneous expectations, even if this assymetric information might yield to multiple equilibria.

4.1.3 Agent-based models, from micro to macro

In macro-economic models, agents' behavior should be in line with the solutions of dynamic optimization problems. Agents are assumed to form expectations and solve complex problems. Reinforcement learning does not require from the agents to use sophisticated reasoning and to compute accurate expectations. Inspired by Holland (1975), Lettau and Uhlig (1999) compare dynamic programming problems and the asymptotic behavior of some simple algorithm that can be seen as reinforcement learning. Lettau and Uhlig (1999) describe the limiting behavior and prove that it converges to the values given by the solution to the Bellman equation.

Gibson (2007) used agent-based macroeconomics to describe labor mobility, with agents willing to experiment in order to learn, as in reinforcement learning algorithms, as in Sinitskaya and Tesfatsion (2015). Dilaver et al. (2018) goes even further, suggesting that Agent-based computational models can provide an alternative to dynamic stochastic general equilibrium (DSGE) models. Börgers et al. (2004), proves decision-makers, with simple learning rules, that obtain feedback information, behave in suggested in standard models developed in dynamic evolutionary game theory. Finally, Rustichini (1999) provides a very interesting discussion about feedback and available information. It compares the case of individuals learning in isolation, having partial information, and the case of social learning, where agents have full information. The convergence to optimal actions depends on procedures used (linear and exponential procedures do not exhibit the same properties).

4.1.4 Single firm dynamics

Jovanovic (1982) gave the framework for most models dealing with industry dynamics with Bayesian learning. In a model of competition between firms with multiple equilibrium, firms are engaged in an adaptive process, where they learn how to play an equilibrium of the game, as in Fudenberg and Levine (1998). In those models, firms know the model that describes the environment, but there are uncertainties. So agents will learn over time about these elements, when new information arrives. Note that this approach is different from the one in evolutionary game theory (as in Samuelson (1997)) for instance, where agents might not even know that they play a game.

Consider a monopolistic firm, taking actions $a_t \in \mathcal{A}$ – say investment decisions – in order to maximize its expected discounted inter-temporal profit. States of the world are $s_t \in \mathcal{S}$, and we assume that they can be modeled via a Markov process. If future investments are uncertain, it can be assumed that the first will use the same optimal decision rule that the one it uses at time t, taking into account available information. Let r_t denote the profit obtained at time t.

In economic literature, rational expectations were usually considered in early models, meaning that the expectation is computed under the true transition probability. Nevertheless, Cyert and DeGroot (1974) or Feldman (1987) suggested that the first should learn this transition probability π, and a Bayesian framework was considered. Starting from a prior belief, transition probabilities T are supposed to belong to some space $\mathcal{T}$, and experience is used to update mixing probabilities on $\mathcal{T}$. Sargent (1993) considered a weaker updating rule, simpler (related to linear approximations in Bayesian models) but not optimal, usually called *adaptative learning*. In that case, belief at time t, $T_t(s, a, s')$ is a weighted sum of $T_{t-1}(s, a, s')$ and some distance between $T(s, a, s')$ and (s_{t-1}, a_{t-1}, s_t) (through some kernel function). If the weight related to the new observation is of order $1/t$, *recursive least squares learning* is obtained; if weights are constant, adaptative learning is here faster than standard Bayesian learning, which is usually seen as a good property when there are shocks in the economy.

Erev and Roth (1998) explicitly introduced the idea of *stock of reinforcement*, corresponding to the standard Q-function. and for any action-state pair (a, s), the updating rule is

$$Q_{t+1}(a, s) \leftarrow Q_t(a, s) + \gamma_t k\big((a, s) - (a_t, s_t)\big)$$

where some kernel k is considered. Recently, Ito and Reguant (2016) used reinforcement learning to describe sequential energy markets.

4.1.5 Adaptative design for experiments

Most experiments are designed to inform about the impact of choosing a policy, among various that can be considered. And more precisely, as discussed in Kasy and Sautmann (2019), the question *which program will have the largest effect* is usually preferred to the question *does this program have a significant effect*, in many cases, see (Chattopadhyay and Duflo 2004) and more recently (Athey and Imbens 2016), and references therein. If dynamic experiments are considered, there are usually several waves, and the optimal experimental design would usually learn from earlier waves, and assign more experimental agents to the better-performing treatments in future waves. Thus, this policy choice problem is a finite-horizon dynamic stochastic optimization problem. Thompson (1933) introduced this idea of adaptive treatment assignment, and Weber (1992) proved that this problem can be expressed using multi-armed bandits, and the optimal solution to this bandit problem is to choose the arm with the to the highest Gittins index, that can be related to the so-called Thompson sampling strategy. Thompson sampling simply assigns the next wave of agents to each treatment with frequencies proportional to the probability that that each treatment is the optimal one.

As explained in Kasy and Sautmann (2019), standard experimental designs are geared toward point estimation and hypothesis testing. But they consider the problem of treatment assignment in an experiment with several non-overlapping waves, where the goal is to choose among a set of possible policies (here treatments). The optimal experimental design learns from earlier waves, and assigns more experimental units to the better-performing treatments in later waves : assignment probabilities are an increasing concave function of the posterior probabilities that each treatment is optimal. They provide theoretical results to this *exploration sampling* design.

4.2 Applications in Operations Research and Game Theory

Probably more interesting is the case where there are multiple strategic agents, interacting (see Zhang et al. 2019 for a nice survey). But before, let us mention the use of reinforcement learning techniques in operation research, and graphs.

4.2.1 Traveling Salesman

A graph (E, V) is a collection of edges E (possibly oriented, possibly weighted) and vertices (or nodes) V. There are many several classical optimization problems on graphs. In the traveling salesman problem, we want to find a subgraph $(E^\star, V)$ (with $E^\star \subset E$) which forms a cycle of minimum total weight that visits each node V at least once. But one might also think of max-flow or max-cut problems, or optimal matching on bipartite graphs (see Galichon 2017 for more examples, with economic applications). In several problems, we seek an optimal solution, which can be a subset $V^\star$ or $E^\star$, of vertices or edges. In the traveling salesman problem (TSP), given an order list of nodes V' that defines a cycle ($E^\star \subset E$ is the subset of edges $\{(V'_i, V'_{i+1})\}$ with $V \subset V$ and $V'_i, V'_{i+1} \in E$ for all i), the associated loss function is

$$\ell(V') = \sum_{i \in |V'|} \left(w(V'_i, V'_{i+1})\right), \text{ with } V'_{|V'|+1} = V_1.$$

Most TSP algorithms are sequential, which will make reinforcement learning perfectly appropriate here. For instance, the 2-opt algorithm (developed in Flood (1956) and Croes (1958)) suggests to iteratively remove two edges and replace these with two different edges that reconnect the fragments created by edge removal into a shorter tour (or that increases the tour least),

$$(i^\star, j^\star) = \underset{i,j=1,\ldots,|V'|}{\text{argmin}} \left\{ \ell(V') - \ell(\tilde{V}^{(ij)}) \right\}, \text{ where } \tilde{V}_k^{(ij)} = \begin{cases} V'_j & \text{if } k = i \\ V'_i & \text{if } k = j \\ V'_k & \text{otherwise} \end{cases}$$

Other popular techniques are for instance Christophides algorithm (developed in Christofides (1976)) or some evolutionary model inspired by ant colonies (as developed in Dorigo and Gambardella (1996)). Here also, it can be interesting to explore possibly non-optimal moves on a short term basis (in the sense that locally they end-up in a longer route) Such sequential techniques can be formulated using the framework of reinforcement learning. The states $\mathcal{S}$ are subsets of edges E in the context of TSP that form a cycle. In the 2-opt algorithm, actions $\mathcal{A}$ are nodes that will be permuted. Rewards are related to changes in the loss function (and the non-discounted sum of rewards is considered here). The nearest neighbour algorithm (which is a greedy algorithm) or cheapest insertion (as defined in Rosenkrantz et al. (1974)) can also be seen with a reinforcement learning algorithm. States $\mathcal{S}$ are subsets of edges E that form partial cycles, and the action $\mathcal{A}$ means growing the route with one node, by inserting it optimally. The rewards is related to the change in the tour length. That idea was developed in Gambardella and Dorigo (1995) recently, or Dai et al. (2017) for a recent survey of reinforcement learning techniques in the context of optimization over graphs.

Deudon et al. (2018) provides insights on how efficient machine learning algorithms could be adapted to solve combinatorial optimization problems in conjunction with existing heuristic procedures. In Bello et al. (2016) the heuristic procedure is replaced by some neural networks. Despite the computational expense, an efficient algorithm is obtained.

4.2.2 Stochastic Games and Equilibrium

Consider n players, each of them taking actions $a_i \in \mathcal{A}_i$ and receives a reward r_i. Let $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathcal{A}$ and $\boldsymbol{r} = (r_1, \ldots, r_n)$. Note that r_i is defined on $\mathcal{S} \times \mathcal{A}$. When $\mathcal{S}$ is a singleton (and there is no uncertainty), it is a simple repeated game (or matrix game). A policy π_i maps $\mathcal{S}$ into $\mathcal{A}_i$. Let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$, and $\boldsymbol{\pi}_{-i}$ the collection of all component policies. Thus, $\boldsymbol{\pi} = (\pi_i, \boldsymbol{\pi}_{-i})$ means that player i uses policy π_i while competitors follow $\boldsymbol{\pi}_{-i}$.

Maskin and Tirole (1988a) introduced the concept of *Markov perfect equilibrium*, which is a set of Markovian policies $\boldsymbol{\pi} =$ which simultaneously forms a Nash equilibrium, as discussed in details in Horst (2005) or Escobar (2013). The existence results of such equilibrium are usually performed in two step: first, we should prove that given any policies chosen by opponents, $\boldsymbol{\pi}_{-i}$, there is a unique solution $V_i^\star(s)$; and then we prove that the static game has a Nash equilibrium for any state s. For the first step, the set of best response for player i is $\Pi_i(\boldsymbol{\pi}_{-i})$ such that $\pi_i^\star \in \Pi_i(\boldsymbol{\pi}_{-i})$ if and only if for any π_i and $s \in \mathcal{S}$, $V_i^{(\pi_i^\star, \boldsymbol{\pi}_{-i})}(s) \geq V_i^{(\pi_i, \boldsymbol{\pi}_{-i})}(s)$. And a Nash equilibrium is a collection of policies $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ such that for each player i, $\pi_i \in \Pi_i(\boldsymbol{\pi}_{-i})$. And therefore, no player can do better when changing policies, when other players continue to use their own strategies.

Littman (1994) used Q-learning algorithms for zero-sum stochastic games, with two players. More precisely,

$$V_1(s) = \max_{\pi}\left\{\min_{a_2\in\mathcal{A}_2}\left\{\sum_{a_1\in\mathcal{A}_1}\pi(s,a_1)Q_1(s,\boldsymbol{a})\right\}\right\} = -V_2(s).$$

Erev and Roth (1998) proved that in many games, a one-parameter reinforcement learning model robustly outperforms the equilibrium predictions. Predictive power is improved by adding a *forgetting* property and valuing *experimentation*, with strong connections with rationality concepts. In the context of games, Franke (2003) applies the approach of reinforcement learning to Arthur (1994)'s El Farol problem, where repeatedly a population of agents decides to go to a bar or stay home, and going is enjoyable if, and only if, the bar is not crowded.

The main difficulty arising when several agents are learning simultaneously in a game is that, for each player, the strategy of all the other players becomes part of the environment. Hence the environment dynamics do not remain stationary as the other players are learning as they play. In such context, classical single agent based reinforcement learning algorithms may not converge to a targeted Nash equilibrium, and typically cycles in between several of them, see Hart and Mas-Colell (2003). As observed by Erev and Roth (1998) or in a more general setting by Perolat et al. (2018), stabilizing procedures such as fictitious play (Robinson 1951) allows to reach Nash equilibria in some (but not all, Shapley (1964)) multi Agent learning setting. Elie et al. (2020) observed that such property also extends to the asymptotic mean field game setting introduced by Huang et al. (2006) and Lasry and Lions (2006a, 2006b), where the size of the population is infinite and shares mean field interaction. Multi-Agent reinforcement learning algorithms still lack scalability when the number of agents becomes large, a weakness that mean field games asymptotic properties may hopefully allow to partially overcome.

4.2.3 Auctions and real-time bidding

The majority of online display ads are served through real-time bidding. To place an ad automatically, and optimally, it is critical for advertisers to have a learning algorithm that cleverly bids. Schwind (2007) did show that seeing the bid decision process as a reinforcement learning problem, where the state space is represented by the auction information and the campaign's real-time parameters, while an action is the bid price to set, was very promising. More recently, Even Dar et al. (2009), Zhang et al. (2014), Cai et al. (2017) or Zhao et al. (2018) use reinforcement learning algorithms to design a bidding strategy.

As pointed out by recent articles, the scalability problem from the large real-world auction volume, and campaign budget, is well handled by state value approximation using neural networks. Dütting et al. (2017) and Feng et al. (2018) suggested to use deep reinforcement learning (with deep neural networks) for the automated design of optimal auctions. Even if the optimal mechanism is unknown, they obtain very efficient algorithm, that outperforms more classical ones.

4.2.4 Oligopoly and Dynamic Games

As in the monopolistic case, the profit of firm i will depend on its investing strategies $a_{i,t}$, the capital of firm i as well as competitors. Models of oligopoly with investment and firm entry and exit have been studied in Ericson and Pakes (1995). And in that framework, multiple equilibira are commonly observed, as proved in Doraszelski and Satterthwaite (2010). The concept of *experience-based equilibrium* was introduced in Fershtman and Pakes (2012), with possibly asymmetric information. Hence, firms use past payoffs to reinforce the probability of choosing an action. In that framework, agents explicitly construct beliefs, which is no longer necessary with reinforcement learning.

With adaptative learning, Marcet and Sargent (1989a, 1989b) proved that there was convergence to a rational expectations equilibrium. The reinforcement learning model is here similar to the previous one, there are no assumption about belief of opponents' strategies. Somehow, those algorithms are more related to evolutionary games. Brown (1951) suggested that firms could form beliefs about competitors' choice probabilities, using some *fictitious plays*, also called *Cournot learnning* (studied more deeply in Hopkins (2002)). Bernheim (1984) and Pearce (1984) added assumptions on firms beliefs, called *rationalizability*, under which we can end-up with Nash equilibria.

Maskin and Tirole (1988a, 1988b) considered the case where two firms compete in a Stackelberg competition: they alternate in moving, and then commit to a price for two periods, before (possibly) adjusting. They did observe cycles and tacit collusion within the two firms. Such a result was confirmed by Kimbrough and Murphy (2008) and Waltman and Kaymak (2008). The later studied repeated Cournot games where all players act simultaneously. They study the use of Q-learning for modeling the learning behavior of firms in that repeated Cournot oligopoly games, and they show that Q-learning firms generally learn to collude with each other, although full collusion usually does not emerge. Such a behavior was also observed in Schwalbe (2019) where self-learning price-setting algorithms can coordinate their pricing behavior to achieve a collusive outcome that maximizes the joint profits of the firms using them.

4.3 Applications in Finance

The dynamic control or hedge of risks on financial markets is a natural playground for the use of reinforcement learning algorithms. In the literature, dynamic risk management problems have been extensively studied in model-driven settings, using the tools from dynamic programming either in continuous or discrete time. In such framework, reinforcement learning algorithms naturally opens the door to innovative model-free numerical approximation schemes for hedging strategies, as soon as a realistic financial market simulator is available. Such simulator may typically incorporate market imperfections and frictions (transaction costs, market impact, liquidity issues...). In the following sections, we detail more specifically recent applications on three topics of interest in such context: pricing and hedging of financial derivatives, optimal asset allocation and market impact modeling.

4.3.1 Risk Management

The valuation and hedging of financial derivatives are usually tackled in the quantitative finance literature using model-driven decision rules in a stochastic environment. Namely, for given model dynamics of the assets on a financial market, pricing and hedging of a derivative boils down to solving a dynamic optimal control problem for a well chosen arbitrage free martingale measure. The practical hedging strategy then makes use of the so-called Greeks, the sensitivities of the risk valuation to the different parameters of the model.

Such analysis usually lacks efficient numerical approximation methods in high dimensional settings, as well as precise tractable analytical solutions in the presence of realistic market frictions or imperfections. In the spirit of Weinan et al. (2017) , Buehler et al. (2019) introduced the idea of using reinforcement learning based algorithm in such context, see also (Fécamp et al. 2019). Let consider given a realistic simulator of the financial market possible trajectories. We can encompass the price and/or hedging strategy of the financial derivative in a neural deep network (or any other approximating class of function), and train/estimate the approximating function in a dynamic way. At each iteration, we measure the empirical performance (i.e. loss) of the hedging strategy obtained on a large number of Monte Carlo simulations, and update its parameters dynamically using any typical reinforcement learning algorithm. In particular, such approach allows to encompass scalable high dimensional risk dynamics as well as realistic market frictions or hedging using a large number of financial derivatives.

The design of the market simulator of course requires model-driven assumptions, such as the choice of a particular class of volatility models, as well as its calibration. Nevertheless, we can mention recent attempts on the design of model free financial market simulator based on generative methods, such as the one developed e.g. in Wiese et al. (2019a, 2019b).

4.3.2 Portfolio Allocation

In a similar manner, the design of dynamic optimal investment strategy naturally falls into the scope of reinforcement learning type algorithms. Such observation goes back to Moody and Saffell (2001) and has developed a growing interest in the recent literature (Deng et al. 2016; Almahdi and Yang 2017): Classical Mean-variance trade-off in a continuous time setting is for example revisited in Wang and Zhou (2019) using such viewpoint. Being given a financial market simulator together with choices of return and risk measurement methods written in terms of running or terminal rewards, one can learn optimal investment strategies using typical reinforcement learning algorithms.

One could argue that such algorithms for portfolio allocation may often be reduced to less sophisticate online or bandit type learning algorithms (Li and Hoi 2014). Such argumentation does not remain valid in the more realistic cases where the investor has a significant impact on the financial assets dynamics, as discussed in the next section.

4.3.3 Market Microstructure

When trades occur at a very high frequency or concern a large volume of shares, buying and selling orders have an impact on the financial market evolution, that one can not neglect. It modifies the shape of the order book, containing the list of waiting orders chosen by the other traders of the market. Being given a realistic order book dynamics simulator (or using the financial market as such), one can optimize using Reinforcement Learning algorithms the dynamic use of market and limit orders, see (Spooner et al. 2018; Guéant and Manziuk 2020; Baldacci et al. 2019). The environment is given by the current order book shapes while the state typically represents the inventory of the trader, on a possibly high-dimensional financial market.

Such framework is with no doubt a perfect fit for reinforcement learning algorithms. Nevertheless, a finer modeling perspective should take into account that the order book dynamics result from the aggregation of other traders actions, i.e. buy or sell orders. Hence, as observed e.g. in Ganesh et al. (2019); Vyetrenko and Xu (2019), such setting is more precisely described as a multi-agent learning problem, as the one described above in Sect. 4.2.2.

The practical use of reinforcement based learning algorithms on financial markets suffers two main drawbacks. The first one is the difficulty to create a realistic financial market simulator, together with the necessity to create a robust optimal trading strategy, in response to the differences between the real market and the virtual one. The second and main one is the lack of stationarity of the financial dynamics, which hereby do not allow to apply efficiently on future market dynamics, the investment strategies learned on the past market data points. Besides, the aggregate use of model-free approaches combined with hardly interpretable black box output policy shall inevitably lead to hardly controllable financial market dynamics.

5 Conclusion

Deep Reinforcement learning is nowadays the most popular technique for (artificial) agent to learn closely optimal strategy by experience. Majors companies are training self driving cars using reinforcement learning (see Folkers et al. 2019, or Kiran et al. 2020 for a state-of-the-art). Such techniques are extremely powerful to models behaviors of animals, consumers, investors, etc. Economists have laid the groundwork for this literature, but computational difficulties slowed them down. Recent advances in computational science are extremely promising, and complex economic or financial problems would benefit from being reviewed in the light of these new results.

Nevertheless, algorithms perform well assuming that a lot of information is available. More importantly, as the exploration may represent a very large number of possibilities, the use of deep reinforcement learning algorithms rapidly requires very important computer power. In finance, despite the lack of stationary of the market, it is worth noting that these algorithms begin to be quite popular.

Funding Arthur Charpentier acknowledges the financial support of the AXA Research Fund through the joint research initiative Use and value of unusual data in actuarial science, as well as NSERC grant 2019-07077.

References

Abbeel, P., & Ng, A. (2004). Apprenticeship learning via inverse reinforcement learning. In Proceedings of the 21st International Conference in Machine Learning (ICML 2004).

Abel, D. (2019). Concepts in Bounded Rationality: Perspectives from Reinforcement Learning. PhD thesis, Brown University.

Aguirregabiria, V., & Mira, P. (2002). Swapping the nested fixed point algorithm: a class of estimators for discrete markov decision models. *Econometrica, 70*(4), 1519–1543.

Aguirregabiria, V., & Mira, P. (2010). Dynamic discrete choice structural models: a survey. *Journal of Econometrics, 156*(1), 38–67.

Almahdi, S., & Yang, S. Y. (2017). An adaptive portfolio trading system: a risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications, 87,* 267–279.

Arthur, W. B. (1991). Designing economic agents that act like human agents: a behavioral approach to bounded rationality. *The American Economic Review, 81*(2), 353–359.

Arthur, W. B. (1994). Inductive reasoning and bounded rationality. *The American Economic Review, 84*(2), 406–411.

Athey, S., & Imbens, G. W. (2016). The econometrics of randomized experiments. ArXiv e-prints.

Athey, S., & Imbens, G. W. (2019). Machine learning methods that economists should know about. *Annual Review of Economics, 11*(1), 685–725.

Aumann, R. J. (1997). Rationality and bounded rationality. *Games and Economic Behavior, 21*(1), 2–14.

Bain, M., & Sammut, C. (1995). A framework for behavioural cloning. In Machine Intelligence 15.

Baldacci, B. Manziuk, I., Mastrolia, T., & Rosenbaum, M. (2019). Market making and incentives design in the presence of a dark pool: a deep reinforcement learning approach. arXiv preprint arXiv:1912.01129.

Barto,A. G., & Singh, S. P. (1991). On the computational economics of reinforcement learning. In D. S. Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton (eds), Connectionist Models, pp. 35 – 44. Morgan Kaufmann.

Basci, E. (1999). Learning by imitation. *Journal of Economic Dynamics and Control, 23*(9), 1569–1585.

Bellman, R. (1957). *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.

Bergemann, D., & Hege, U. (1998). Venture capital financing, moral hazard and learning. *Journal of Banking and Finance, 22*(6), 703–735.

Bergemann, D., & Hege, U. (2005). The financing of innovation: Learning and stopping. *The RAND Journal of Economics, 36*(4), 719–752.

Bergemann, D., & Välimäki, J. (1996). Learning and strategic pricing. *Econometrica, 64*(5), 1125–1149.

Bernheim, B. D. (1984). Rationalizable strategic behavior. *Econometrica, 52*(4), 1007–1028.

Berry, D. A., & Fristedt, B. (1985). Bandits Problems Sequential Allocation of Experiments. — (Monographs on statistics and applied probability). Chapman and Hall.

Bertsekas, D. P., & Tsitsiklis, J. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Bottou, L. (1998). Online algorithms and stochastic approximations. In D. Saad (ed), Online Learning and Neural Networks.

Brown, G. W. (1951). Iterative solutions of games by fictitious play. In T. Koopmans (Ed.), *Activity Analysis of Production and Allocation* (pp. 374–376). NewYork: Wiley.

Buehler, H., Gonon, L., Teichmann, J., & Wood, B. (2019). Deep hedging. *Quantitative Finance, 19*(8), 1271–1291.

Börgers, T., Morales, A. J., & Sarin, R. (2004). Expedient and monotone learning rules. *Econometrica, 72*(2), 383–405.

Cai, H., Ren, K., Zhang, W., Malialis, K., Wang, J., Yu, Y., & Guo, D. (2017). Real-time bidding by reinforcement learning in display advertising. In Proceedings of the Tenth ACM International

Conference on Web Search and Data Mining, WSDM '17, pp. 661–670. Association for Computing Machinery: New York, USA.

Charpentier, A., Flachaire, E., & Ly, A. (2018). Econometrics and machine learning. *Economics and Statistics, 505*(1), 147–169.

Chattopadhyay, R., & Duflo, E. (2004). Women as policy makers: Evidence from a randomized policy experiment in india. *Econometrica, 72*(5), 1409–1443.

Cherniak, C. (1986). *Minimal Rationality*. MIT Press: MIT Press.

Christofides, N. (1976). *Worst-case analysis of a new heuristic for the travelling salesman problem*. Graduate School of Industrial Administration, CMU: Technical report.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations research, 6*(6), 791–812.

Cyert, R. M., & DeGroot, M. H. (1974). Rational expectations and bayesian analysis. *Journal of Political Economy, 82*(3), 521–536.

Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. arXiv preprint arXiv:1704.01665.

Deng, Y., Bao, F., Kong, Y., Ren, Z., & Dai, Q. (2016). Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems, 28*(3), 653–664.

Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y., & Rousseau, L.-M. (2018). Learning heuristics for the tsp by policy gradient. Artificial Intelligence, and Operations Research. In W.-J. van Hoeve (Ed.), *Integration of Constraint Programming* (pp. 170–181). Cham: Springer International Publishing.

Devaine, M., Gaillard, P., Goude, Y., & Stoltz, G. (2013). Forecasting electricity consumption by aggregating specialized experts. *Machine Learning, 90*(2), 231–260.

Dilaver, O., Calvert Jump, R., & Levine, P. (2018). Agent-based macroeconomics and dynamic stochastic general equilibrium models: Where do we go from here? *Journal of Economic Surveys, 32*(4), 1134–1159.

Doraszelski, U., & Satterthwaite, M. (2010). Computable markov-perfect industry dynamics. *The RAND Journal of Economics, 41*(2), 215–243.

Dorigo,M., & Gambardella, L. M. (1996). Ant colonies for the traveling salesman problem. Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, 3.

Dütting, P., Feng, Z., Narasimhan, H., Parkes, D. C., & Ravindranath, S. S. (2017). Optimal auctions through deep learning.

Elie, R., Perolat, J., Laurière, M., Geist, M., & Pietquin, O. (2020). On the convergence of model free learning in mean field games. In AAAI Conference one Artificial Intelligence (AAAI 2020).

Erev, I., & Roth, A. E. (1998). Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *American Economic Review, 88*(4), 848–881.

Ericson, R., & Pakes, A. (1995). Markov-perfect industry dynamics: a framework for empirical work. *The Review of Economic Studies, 62*(1), 53–82.

Escobar, J. F. (2013). Equilibrium analysis of dynamic models of imperfect competition. *International Journal of Industrial Organization, 31*(1), 92–101.

Even Dar, E., Mirrokni, V. S., Muthukrishnan, S., Mansour, Y., & Nadav, U. (2009). Bid optimization for broad match ad auctions. In Proceedings of the 18th International Conference on World Wide Web, WWW '09, pages 231–240. Association for Computing Machinery: New York, USA.

Feldman, M. (1987). Bayesian learning and convergence to rational expectations. *Journal of Mathematical Economics, 16*(3), 297–313.

Feng, Z., Narasimhan, H., Parkes, D. C. (2018). Deep learning for revenue-optimal auctions with budgets. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '18, pp. 354–362. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC.

Fershtman, C., & Pakes, A. (2012). Dynamic Games with Asymmetric Information: a Framework for Empirical Work*. *The Quarterly Journal of Economics, 127*(4), 1611–1661.

Flood, M. M. (1956). The travelling salesman problem. *Operations Research, 4,* 61–75.

Folkers, A., Rick, M., & Buskens, C. (2019). Controlling an autonomous vehicle with deep reinforcement learning. 2019 IEEE Intelligent Vehicles Symposium (IV). https://doi.org/10.1109/ivs.2019.8814124.

Franke, R. (2003). Reinforcement learning in the el farol model. *Journal of Economic Behavior and Organization, 51*(3), 367–388.

Fudenberg, D., & Levine, D. (1998). *The Theory of Learning in Games*. USA: Massachusetts Institute of Technology (MIT) Press.

Fécamp, S., Mikael, J., & Warin, X. (2019). Risk management with machine-learning-based algorithms. arXiv preprint arXiv:1902.05287,.

Gabaix, X. (2014). A sparsity-based model of bounded rationality. *The Quarterly Journal of Economics, 129*(4), 1661–1710.

Galichon, A. (2017). *Optimal transport methods in economics*. USA: Princeton University Press.

Gambardella, L. M., & Dorigo, M. (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem. In A. Prieditis and S. Russell, editors, Machine Learning Proceedings 1995, pp. 252–260. Morgan Kaufmann.

Ganesh, S., Vadori, N., Xu, M., Zheng, H., Reddy, P., & Veloso, M. (2019). Reinforcement learning for market making in a multi-agent dealer market. arXiv preprint arXiv:1911.05892.

Garcia, J. (1981). The nature of learning explanations. *Behavioral and Brain Sciences, 4*(1), 143–144.

Gennaioli, N., & Shleifer, A. (2010). What Comes to Mind*. *The Quarterly Journal of Economics, 125*(4), 1399–1433.

Gershman, S. J., Horvitz, E. J., & Tenenbaum, J. B. (2015). Computational rationality: a converging paradigm for intelligence in brains, minds, and machines. *Science, 349*(6245), 273–278.

Gibson, B. (2007). A multi-agent systems approach to microeconomic foundations of macro. Economics Department Working Paper, University of Massachusetts, 2007-10.

Gigerenzer, G., & Goldstein, D. (1996). Reasoning the fast and frugal way: models of bounded rationality. *Psychological review, 103*(4), 650.

Gittins, J. (1989). *Bandit processes and dynamic allocation indices*. NewYork: Wiley.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org.

Granato, J., Guse, E. A., & Wong, M. C. S. (2008). Learning from the expectations of others. *Macroeconomic Dynamics, 12*(3), 345–377. https://doi.org/10.1017/S1365100507070186.

Guéant, O., & Manziuk, I. (2020). Deep reinforcement learning for market making in corporate bonds: beating the curse of dimensionality. *Applied Mathematical Finance, 26*(5), 387–452.

Hansen, L. P., & Sargent, T. J. (2013). Recursive Models of Dynamic Linear Economies. The Gorman Lectures in Economics. Princeton University Press.

Hart, S., & Mas-Colell, A. (2003). Uncoupled dynamics do not lead to nash equilibrium. *American Economic Review, 93*(5), 1830–1836.

Hasselt, H. V. (2010). Double q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, Advances in Neural Information Processing Systems 23, pp. 2613–2621. Curran Associates, Inc.

Hellwig, M. F. (1973). Sequential models in economic dynamics. PhD thesis, Massachusetts Institute of Technology, Department of Economics.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. USA: University of Michigan Press.

Holland, J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Hopkins, E. (2002). Two competing models of how people learn in games. *Econometrica, 70*(6), 2141–2166.

Horst, U. (2005). Stationary equilibria in discounted stochastic games with weakly interacting players. *Games and Economic Behavior, 51*(1), 83–108.

Hotz, V. J., & Miller, R. A. (1993). Conditional choice probabilities and the estimation of dynamic models. *The Review of Economic Studies, 60*(3), 497–529.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. Cambridge, Massachusetts: MIT Press.

Huang, M., Malhamé, R. P., & Caines, P. E. (2006). Large population stochastic dynamic games: closed-loop mckean-vlasov systems and the nash certainty equivalence principle. *Communications in Information and Systems, 6*(3), 221–252.

Hughes, N. (2014). Applying reinforcement learning to economic problems. Technical report, Australian National University.

Igami, M. (2017). Artificial intelligence as structural estimation: Economic interpretations of deep blue, bonanza, and alphago. arXiv preprint arXiv:1710.10967.

Ito, K., & Reguant, M. (2016). Sequential markets, market power, and arbitrage. *American Economic Review, 106*(7), 1921–57. https://doi.org/10.1257/aer.20141529.

Jenkins, H. M. (1979). Animal learning and behavior theory. In E. Hearst (ed), The first century of experimental psychology, pp. 177–228.

Jovanovic, B. (1982). Selection and the evolution of industry. *Econometrica, 50*(3), 649–670.

Kahneman, D. (2011). *Thinking, fast and slow*. NewYork: Macmillan.

Kasy, M., & Sautmann, A. (2019). Adaptive treatment assignment in experiments for policy choice. Technical report, Harvard University.

Keller, G., & Rady, S. (1999). Optimal experimentation in a changing environment. *The Review of Economic Studies, 66*(3), 475–507.

Kimbrough, S. O., & Murphy, F. H. (2008). Learning to collude tacitly on production levels by oligopolistic agents. *Computational Economics, 33*(1), 47.

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A. A., Yogamani, S., & Pérez, P. (2020). Deep reinforcement learning for autonomous driving: A survey. arXiv preprint arXiv:2002.00444, 2020.

Kiyotaki, N., & Wright, R. (1989). On money as a medium of exchange. *Journal of Political Economy, 97*(4), 927–954.

Klein, E., Geist, M., Piot, B., & Pietquin, O. (2012). Inverse reinforcement learning through structured classification. *Advances in Neural Information Processing Systems,* 1007–1015.

Lagoudakis, M., & Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research, 4,* 1107–1149.

Lasry, J.-M., & Lions, P.-L. (2006a). Jeux à champ moyen. i - le cas stationnaire. *Comptes Rendus Mathematique, 343*(9), 619–625.

Lasry, J.-M., & Lions, P.-L. (2006b). Jeux à champ moyen. ii - horizon fini et contrôle optimal. *Comptes Rendus Mathematique, 343*(10), 679–684.

Leimar, O., & McNamara, J. (2019). Learning leads to bounded rationality and the evolution of cognitive bias in public goods games. *Nature Scientific Reports, 9,* 16319.

Lettau, M., & Uhlig, H. (1999). Rules of thumb versus dynamic programming. *American Economic Review, 89*(1), 148–174.

Levina, T., Levin, Y., McGill, J., & Nediak, M. (2009). Dynamic pricing with online learning and strategic consumers: an application of the aggregating algorithm. *Operations Research, 57*(2), 327–341.

Li, B., & Hoi, S. C. (2014). Online portfolio selection: a survey. *ACM Computing Surveys (CSUR), 46*(3), 1–36.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In Machine Learning Proceedings 1994, pp. 157–163. Elsevier.

Ljungqvist, L., & Sargent, T. J. (2018). *Recursive macroeconomic theory* (Vol. 4). USA: MIT Press.

Magnac, T., & Thesmar, D. (2002). Identifying dynamic discrete decision processes. *Econometrica, 70*(2), 801–816.

Marcet, A., & Sargent, T. J. (1989a). Convergence of least-squares learning in environments with hidden state variables and private information. *Journal of Political Economy, 97*(6), 1306–1322.

Marcet, A., & Sargent, T. J. (1989b). Convergence of least squares learning mechanisms in self-referential linear stochastic models. *Journal of Economic Theory, 48*(2), 337–368.

Maskin, E., & Tirole, J. (1988a). A theory of dynamic oligopoly, I: Overview and quantity competition with large fixed costs. *Econometrica, 56,* 549–569.

Maskin, E., & Tirole, J. (1988b). A theory of dynamic oligopoly, II: Price competition, kinked demand curves, and edgeworth cycles. *Econometrica, 56,* 571–579.

McLennan, A. (1984). Price dispersion and incomplete learning in the long run. *Journal of Economic Dynamics and Control, 7*(3), 331–347.

Miller, R. A. (1984). Job matching and occupational choice. *Journal of Political Economy, 92*(6), 1086–1120.

Minsky, M. (1961). Steps toward artificial intelligence. *Transactions on Institute of Radio Engineers, 49,* 8–30.

Misra, K., Schwartz, E. M., & Abernethy, J. (2019). Dynamic online pricing with incomplete information using multiarmed bandit experiments. *Marketing Science, 38*(2), 226–252.

Moody, J., & Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks, 12*(4), 875–889.

Mullainathan, S., & Spiess, J. (2017). Machine learning: an applied econometric approach. *Journal of Economic Perspectives, 31*(2), 87–106.

Nedić, A., & Bertsekas, D. P. (2003). Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems, 13,* 79–110.

Ng, A. Y., Russell, S. J., et al. (2000). Algorithms for inverse reinforcement learning. *Proceedings of the International Conference on Machine Learning (ICML),* 663–670.

O'Neill, D., Levorato, M., Goldsmith, A., & Mitra, U. (Oct 2010). Residential demand response using reinforcement learning. In 2010 First IEEE International Conference on Smart Grid Communications, pp. 409–414.

Pakes, A. (1986). Patents as options: some estimates of the value of holding european patent stocks. *Econometrica, 54*(4), 755–784.

Pakes, A., & Schankerman, M. (1984). *The rate of obsolescence of patents, research gestation lags, and the private rate of return to research resources* (pp. 73–88). Chicago: University of Chicago Press.

Pearce, D. G. (1984). Rationalizable strategic behavior and the problem of perfection. *Econometrica, 52*(4), 1029–1050.

Pearl, J. (2019). The seven tools of causal inference, with reflections on machine learning. *Commununications of the ACM, 62*(3), 54–60.

Perolat, J., Piot, B., & Pietquin, O. (2018). Actor-critic fictitious play in simultaneous move multistage games. *International Conference on Artificial Intelligence and Statistics,* pp. 919–928.

Rescorla, R. A. (1979). Aspects of the reinforcer learned in second-order Pavlovian conditioning. *Journal of Experimental Psychology: Animal Behavior Processes, 5*(1), 79–95.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society, 58*(5), 527–535.

Robinson, J. (1951). An iterative method of solving a game. *Annals of mathematics,* 296–301.

Rosenkrantz, D. J., Stearns, R. E., & Lewis, P. M. (Oct 1974). Approximate algorithms for the traveling salesperson problem. In 15th Annual Symposium on Switching and Automata Theory (swat 1974), pp. 33–42.

Rothkopf, C. A., & Dimitrakakis, C. Preference elicitation and inverse reinforcement learning. In Machine Learning and Knowledge Discovery in Databases, pp. 34–48. Springer: Berlin.

Rothschild, M. (1974). A two-armed bandit theory of market pricing. *Journal of Economic Theory, 9*(2), 185–202.

Rubinstein, A. (1998). *Modeling Bounded Rationality*. USA: MIT Press.

Russell, S. J., & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall: New Jersey.

Russell, S. J., & Subramanian, D. (1995). Provably bounded-optimal agents. *Journal of Artificial Intelligence Research, 2*(1), 575–609.

Rust, J. (1987). Optimal replacement of gmc bus engines: An empirical model of harold zurcher. *Econometrica, 55*(5), 999–1033.

Rustichini, A. (1999). Optimal properties of stimulus-response learning models. *Games and Economic Behavior, 29*(1), 244–273. https://doi.org/10.1006/game.1999.0712.

Samuelson, L. (1997). *Evolutionary games and equilibrium selection*. Mass: MIT Press Cambridge.

Sargent, T. (1993). *Bounded rationality in macroeconomics*. Oxford: Oxford University Press.

Schaal, S. (1996). Learning from demonstration. In Proceedings of the 9th International Conference on Neural Information Processing Systems, NIPS'96, pp.1040-1046, Cambridge, MA, USA. MIT Press.

Schwalbe, U. (2019). Algorithms, machine learning, and collusion. *Journal of Competition Law and Economics, 14*(4), 568–607.

Schwind, M. (2007). *Dynamic pricing and automated resource allocation for complex information services: reinforcement learning and combinatorial auctions*. Berlin: Springer-Verlag.

Semenova, V. (2018). Machine learning for dynamic discrete choice. arXiv preprint arXiv:1808.02569.

Shapley, L. (1964). Some topics in two-person games. *Advances in Game Theory, 52,* 1–29.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science, 362*(6419), 1140–1144.

Simon, H. A. (1972). Theories of bounded rationality. *Decision and Organization, 1*(1), 161–176.

Sinitskaya, E., & Tesfatsion, L. (2015). Macroeconomies as constructively rational games. *Journal of Economic Dynamics and Control, 61,* 152–182.

Skinner, B. F. (1938). *The behavior of organisms: an experimental analysis*. New York: Appleton-Century-Crofts.

Spooner, T., Fearnley, J., Savani, R., & Koukorinis, A. (2018). Market making via reinforcement learning. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, pp. 434–442. International Foundation for Autonomous Agents and Multiagent Systems.

Stokey, N. L., Lucas, R. E., & Prescott, E. C. (1989). *Recursive methods in economic dynamics*. Cambridge: Harvard University Press.

Su, C.-L., & Judd, K. L. (2012). Constrained optimization approaches to estimation of structural models. *Econometrica, 80*(5), 2213–2230.

Sutton, R. S., & Barto, A. G. (1981). Toward a modern theory of adaptive networks: expectation and prediction. *Psychological Review, 88*(2), 135.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIP Press.

Tamar, A., Chow, Y., Ghavamzadeh, M., & Mannor, S. (2015). Policy gradient for coherent risk measures. *Advances in Neural Information Processing Systems,* 1468–1476.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika, 25*(3/4), 285–294.

Thorndike, E. L. (1911). *Animal Intelligence*. New York, NY: Macmillan.

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological review, 55*(4), 189.

Vyetrenko, S., & Xu, S. (2019). Risk-sensitive compact decision trees for autonomous execution in presence of simulated market response. arXiv preprint arXiv:1906.02312

Waltman, L., & Kaymak, U. (2008). *q*-learning agents in a cournot oligopoly model. *Journal of Economic Dynamics and Control, 32*(10), 3275–3293.

Wang, H., & Zhou, X.Y. (2019). Continuous-time mean-variance portfolio optimization via reinforcement learning. arXiv preprint arXiv:1904.11392

Watkins, C.J. (1989). Learning from delayed reward. PhD thesis, Cambridge University

Watkins, C. J. C. H., & Dayan, P. (1992). *q*-learning. *Machine Learning, 8*(3), 279–292.

Weber, R. (1992). On the gittins index for multiarmed bandits. *The Annals of Applied Probability, 2*(4), 1024–1033.

Weinan, E., Han, J., & Jentzen, A. (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics, 5*(4), 349–380.

Weitzman, M. L. (1979). Optimal search for the best alternative. *Econometrica, 47*(3), 641–654.

Whittle, P. (1983). *Optimization Over Time* (Vol. 1). Chichester, UK: Wiley.

Wiese, M., Bai, L., Wood, B., & Buehler, H. (2019a). Deep hedging: learning to simulate equity option markets. Available at SSRN 3470756

Wiese, M., Knobloch, R., Korn, R., & Kretschmer, P. (2019b). Quant gans: deep generation of financial time series. arXiv preprint arXiv:1907.06673

Wintenberger, O. (2017). Optimal learning with bernstein online aggregation. *Machine Learning, 106*(1), 119–141.

Wolpin, K. I. (1984). An estimable dynamic stochastic model of fertility and child mortality. *Journal of Political Economy, 92*(5), 852–874.

Zhang, K., Yang, Z., & Başar, T. (2019). Multi-agent reinforcement learning: a selective overview of theories and algorithms

Zhang, W., Yuan, S., & Wang, J. (2014). Optimal real-time bidding for display advertising. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, pp. 1077–1086, New York, NY, USA. Association for Computing Machinery.

Zhao, J., Qiu, G., Guan, Z., Zhao, W., He, X. (2018). Deep reinforcement learning for sponsored search real-time bidding. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '18, pp. 1021–1030. Association for Computing Machinery: New York, USA.