Subchapter 1.1

Reinforcement Learning Methods in Algorithmic Trading

Olivier Guéant[1]

1 – Université Paris 1 Panthéon-Sorbonne, Centre d'Economie de la Sorbonne, 106 Boulevard de l'Hôpital, 75642 Paris Cedex 13, France. Email: olivier.gueant@univ-paris1.fr

This subchapter is dedicated to the third paradigm of machine learning alongside supervised and unsupervised learning: reinforcement learning (RL). RL methods have recently been successful in solving complex dynamic optimization problems in domains such as robotics, video games, and board games. Being flexible in terms of modelling and scalable to high dimensions, they are often regarded as good candidates to solve many financial problems, especially in the field of algorithmic trading. The goal of this subchapter is multifold: presenting the main ideas and concepts of RL, discussing their relevance for addressing algorithmic trading problems, reviewing the existing applications, and discussing the future. In particular, our view is that the range of problems that could be addressed with RL techniques is narrower than what most people think, but that RL-based trading programs could be competitive in execution and market making if traditional quants, computer scientists, and engineers united forces.

1.1.1 Introduction

1.1.1.1 The recent successes of reinforcement learning

Since the middle of the 2010s, all fields of science have been impacted by machine learning techniques. Finance, and in particular algorithmic trading, is, of course, no exception. Many techniques of supervised and unsupervised learning have indeed become fashionable challengers to existing statistical and econometrical techniques and have been tested by both academics and practitioners; sometimes with great success!

Alongside supervised and unsupervised learning, reinforcement learning (RL) – the third machine learning paradigm – also got in the limelight as RL-based computer programs have recently been successful in playing video games and board games. Researchers at DeepMind (see (Mnih et al., 2015)) have indeed built a deep Q-network agent playing a long list of Atari 2600 games at human level or above. What stunned most researchers was that this agent (i) only learned with the pixels and the game score as inputs, and (ii) used a single algorithm, network architecture, and set of hyperparameters for all games. A few months later, another RL-based computer program called AlphaGo made the headlines after defeating several professional Go players. AlphaGo itself was later soundly defeated by AlphaGo Zero. The latter is another RL-agent that learned to master the game of Go from self-play with no initial knowledge but the game rules; and with the position of the stones on the board as its input instead of hand-engineered features.[1] Interestingly, these programs are often regarded as "creative" as they developed unconventional strategies.

Although RL is not a new field, the buzz surrounding its recent successes has led to new research efforts and new hopes in domains as varied as robotics, self-driving cars, healthcare, and, of course, finance.

1.1.1.2 Finance, it might be your go

RL is aimed at solving problems involving an agent interacting with an environment – possibly stochastic – so as to maximize an expected numerical reward. Therefore, there is no surprise that the financial community has recurrently been curious about RL tools.

[1] For more details, see (Silver et al., 2016) and (Silver et al., 2017). See also (Silver et al., 2018) for another version called AlphaZero that learned (at the same time!) to master chess and shōgi as well as Go from self-play. Very recently, a new step forward was made with the MuZero algorithm that learned to play the above board games alongside Atari video games without being told the rules – see (Schrittwieser et al., 2020).

In fact, as we shall see in Section 1.1.4, there has been for at least two decades some research works here and there using RL techniques in the domain of finance, especially in algorithmic trading. However, RL tools have settled over the last couple of years on the forefront of the financial scene. Of course the initial trigger was the successes of DeepMind discussed above, but the new popularity of RL has also been due to (i) the communication of some banks announcing the advent of execution trading robots based on deep RL (see for instance (Noonan, 2017) in the Financial Times), and (ii) the publication of the now famous paper "Deep Hedging" (see (Buehler et al., 2019)).

"Deep Hedging" aimed at showing that it is possible to find the optimal strategy to hedge a European contingent claim in any model thanks to neural networks and a direct policy search algorithm. The authors proposed more precisely recurrent and semi-recurrent neural network architectures in order to approximate the strategy that minimizes the risk borne by a hedger[2] – the risk being measured through several risk measures including CVaR.[3]

Although it only deals with option pricing (in fact option hedging), "Deep Hedging" has influenced the whole quantitative finance community beyond derivatives. First, it has forced a lot of quant practitioners to consider optimization tools with fresh eyes and to recall that pricing derives from hedging – and not the other way round, contrary to what the usual computation of Greeks could let think. Second, in addition to bringing optimization tools in the limelight, the paper has exemplified the flexibility of RL methods based on simulated data.[4] The direct policy search method proposed in "Deep Hedging" can indeed use, for training, data simulated from any model or even from a mix of models.[5] This second point is important as it enlarges the range of models that can be used; and this may be crucial in an industry that increasingly aims to manage model risk. Third, the paper has conveyed the message that RL methods could be used to solve a wide range of high-dimensional problems, far beyond those (often linear) traditionally addressed with Monte-Carlo simulations.[6]

[2]The use of neural networks for the pricing and hedging of options is not a new topic and we refer the interested readers to the thorough review work that was recently carried out in (Ruf and Wang, 2020).

[3]In particular, by using a famous trick due to Rockafellar and Uryasev (see (Rockafellar and Uryasev, 2002)), they have showed that CVaR is a great risk measure for some RL models that need to account for risk (at least for problems solved using a direct policy search approach).

[4]We intentionally avoid the use of the expression "model-free" because it is ambiguous, at least for financial applications (see also Section 1.1.3).

[5]Historical data can also be used but RL methods usually require more data points than what historical data can provide (see Section 1.1.3 for more details).

[6]In recent years, in parallel to this renewed interest for RL as a way to approximate the optimal solution of high-dimensional stochastic optimal control problems, some other approaches have been proposed, in particular to numerically approximate the solutions of Hamilton-Jacobi-Bellman equations in high dimension. In particular, in a series of papers including for instance (E et al., 2017) and (Han et al., 2018), a group of researchers used the representation of a linear

In this context, this subchapter aims to discuss the interest of RL techniques for algorithmic trading. We start by presenting the main concepts and ideas traditionally associated with RL. We then highlight the numerous differences between (i) most of the toy examples of the RL community or even video games and board games, and (ii) the real-life problems of algorithmic trading. Further, we review the existing research works applying RL ideas to algorithmic trading problems.[7] Finally, we discuss future perspectives and insist on the fact that, if the quantitative finance community wishes to see RL algorithms implemented on a large scale, then the involvement of computer scientists and engineers is of utmost importance.

1.1.2 A brief introduction to reinforcement learning

RL encompasses a wide range of methods aimed at maximizing the expected reward of an agent interacting with a deterministic or stochastic environment. In quantitative finance, this type of problems is often addressed by using the mathematical tools of deterministic and stochastic optimal control. Most techniques of optimal control, in particular those based on the dynamic programming principle, are part of RL techniques, but RL methods often go beyond optimal control / dynamic programming methods in at least two aspects. First, many RL methods are not based on grids,[8] but instead on function approximations. Therefore, they do not suffer (or at least suffer less) from the curse of dimensionality.[9] Second, many RL techniques use data samples and do not require to know the transition kernel (that defines the dynamics of the state variables and the distribution of the rewards).

---

or nonlinear parabolic partial differential equation (PDE) with a backward stochastic differential equation (BSDE) in order to build what they call a BSDE solver that approximates the solution of the PDE and – in fact, through – its gradient. In addition to the above papers, we refer to (Becker et al., 2019), (Henry-Labordere, 2017), (Huré et al., 2019) and (Pham et al., 2019) for additional discussions and extensions, especially to the case of optimal stopping problems leading to variational inequalities or more complex PDE. Some authors classify these methods as RL because of (i) the use of machine learning techniques (neural networks, stochastic gradient descent, etc.), and (ii) the central use of the gradient of the solution of the PDE, which, in the case of a Hamilton-Jacobi-Bellman equation, is intimately related to the optimal control (or optimal action in the vocabulary of RL). This classification is in fact questionable and we do not consider these approaches RL ones. In particular, it should be noted that these methods do not "explore" unlike many RL methods.

[7] The readers with wider economic or financial interests can read the recent reviews carried out in (Charpentier et al., 2020), (Fischer, 2018), and (Kolm and Ritter, 2020).

[8] The usual grid methods based on the dynamic programming principle are part of the so-called tabular methods in RL.

[9] In order to beat the curse of dimensionality, it is possible, when the dimension remains reasonable, to use quantization methods (see (Pagès et al., 2004)). This is for instance what was done in the paper (Abergel et al., 2020) dealing with market making in a limit order book.

Let us now start our brief introduction to RL. Our goal is not to be exhaustive, but rather to present the main ideas and concepts. Like in many RL seminal references (we recommend in particular (Bertsekas and Tsitsiklis, 1996), (Bertsekas, 2019), (Powell, 2011), (Sutton and Barto, 2018), and (Szepesvári, 2010), and refer to them for a more detailed introduction) we consider a discrete-time approach.[10] We therefore start with Markov Decision Processes (MDP)[11] and present the different types of optimization problems addressed in RL. This presentation is followed by a list of concepts that are commonly employed in the RL literature. Then we present and discuss the different families of RL algorithms.

1.1.2.1 Markov Decision Processes and optimization problems

Formally, a MDP is a triplet $(\mathcal{S}, \mathcal{A}, \mathcal{P})$ where:

- $\mathcal{S}$, called the state space, describes the different states in which the system can be – it is typically either a finite or countable set or a subset of a finite-dimension space;
- $\mathcal{A}$, called the action space, describes the different actions the agent (or decision-maker) can choose – it is typically either a finite or countable set or a subset of a finite-dimension space;
- $\mathcal{P}$ is a probability kernel that maps a couple $(s, a) \in \mathcal{S} \times \mathcal{A}$ to a probability measure $\bar{p}(\cdot, \cdot|s, a)$ on $\mathcal{S} \times \mathbb{R}$ (or $\mathcal{S} \times (\mathbb{R} \cup \{-\infty\})$) where $\bar{p}(\cdot, \cdot|s, a)$ models for a given state s and action a the distribution of the next state and the associated reward.

In practice, it may be more convenient to replace the transition kernel by two concepts. First, a state transition kernel that maps a couple $(s, a) \in \mathcal{S} \times \mathcal{A}$ to a probability measure $p(\cdot|s, a)$ on $\mathcal{S}$ modelling for a given state s and action a the distribution of the next state s'. Second, a probability distribution for the reward given (s, a, s') or, as it is often enough, a reward function r that maps a triplet (s, a, s') of current state, action, and next state to the expected reward, or a variant of it where one takes the expectation over all the possible next states s' given (s, a).

MDP are essential for modelling sequential decision-making problems. Starting from a given state S_0, one can build recursively a sequence $(S_n, A_n, R_{n+1})_n$ of states, actions, and rewards by assigning at date n, once A_n has been chosen, the distribution $p(\cdot|S_n, A_n)$ to S_{n+1} and setting R_{n+1} to the associated reward or its expectation, i.e. to $r(S_n, A_n, S_{n+1})$ or $r(S_n, A_n)$ – we use the latter to simplify in

[10] The readers used to continuous-time optimal control may find it more natural to start with the work of Munos, for instance his *habilitation* (Munos, 2004) and the references therein.

[11] We do not cover the case of Partially Observable MDP (POMDP). We refer to (Bäuerle and Rieder, 2011) for a detailed introduction to MDP (with applications to finance) that covers POMDP.

most of what follows.[12]

Given a MDP, RL methods are aimed at maximizing an objective function or expected score that depends on the choice of actions. Two main types of optimization problems are traditionally considered:[13]

- Finite-horizon problems, in which one maximizes the expected score

$$\mathbb{E}\left[\sum_{n=0}^{N-1} r(S_n, A_n) + R(S_N)\right]$$

  for a given time horizon N and a final payoff function R;

- Infinite-horizon problems, in which one maximizes the expected discounted score

$$\mathbb{E}\left[\sum_{n=0}^{+\infty} \gamma^n r(S_n, A_n)\right],$$

  for a given discount $\gamma \in (0, 1)$.

1.1.2.2 Basic concepts

Several concepts have been introduced in order to address the above dynamic optimization problems:

- *Policy*: a policy is a function that maps a time and a state to an action (in the case of a deterministic policy) or to a probability measure on the action space (in the case of a so-called stochastic policy).[14] We call stationary a policy that does not depend on time.

- *Optimal policy*: an optimal policy is one that maximizes the objective function / expected score. Optimal policies are what RL algorithms ultimately look for.

[12] A specific case plays an important part in the literature: bandit problems, where $p(\cdot|s, a)$ does not depend on a. In that case, the state space is often a singleton, but it can also be more complex in the case of contextual bandits. In any case, the problem is essentially that of a gambler in front of a set of slot machines who needs to choose the best machine to play (in an online manner). We cover in more details RL methods where $p(\cdot|s, a)$ does depend on a, but it is interesting to notice that the classical approaches (see, for instance, (Thompson, 1933) on Thompson sampling and (Auer, 2002; Auer et al., 2002) on the upper confidence bound paradigm) are useful in algorithmic finance, for instance to choose between algorithms. In particular multi-armed bandit methods based on the exploration-exploitation trade-off could be very useful when several execution algorithms with the same benchmark (e.g. VWAP) are available and one needs to determine which one is the best in practice.

[13] Other types of problems do exist, such as infinite-time problems with no discount but an absorbing state, average reward (ergodic) problems, etc.

[14] A deterministic policy is of course a stochastic policy where the probability measure is a Dirac measure.

- *Value function (or state value function)*: value functions map states to expected scores in order to evaluate the performance of a policy. In the case of a finite-horizon problem, the value function V^π associated with a policy π is defined as

$$V^\pi : (k, s) \mapsto \mathbb{E}\left[\sum_{n=k}^{N-1} r(S_n, \pi_n(S_n)) + R(S_N) | S_k = s\right].$$

  In the case of an infinite-horizon problem, the value function V^π associated with a stationary policy π is defined as

$$V^\pi : s \mapsto \mathbb{E}\left[\sum_{n=0}^{+\infty} \gamma^n r(S_n, \pi(S_n)) \middle| S_0 = s\right].$$

- *Optimal value function*: the optimal value function V^* is the value function associated with an optimal policy.[15]

- *State-action value function or Q function*: the state-action value function associated with a policy is a variant of the value function associated with that policy where the first action is prescribed. In the case of a finite-horizon problem, Q^π is defined as[16]

$$Q^\pi : (k, s, a) \mapsto \mathbb{E}\left[r(S_k, A_k) + \sum_{n=k+1}^{N-1} r(S_n, \pi_n(S_n)) + R(S_N) \middle| S_k = s, A_k = a\right].$$

  In the case of an infinite-horizon problem (where it is often used), Q^π is defined as[17]

$$Q^\pi : (s, a) \mapsto \mathbb{E}\left[r(S_0, A_0) + \sum_{n=1}^{+\infty} \gamma^n r(S_n, \pi(S_n)) \middle| S_0 = s, A_0 = a\right].$$

- *Optimal state-action value function or optimal Q function*: the optimal state-action value function Q^* is the state-action value function associated with an optimal policy.[18]

- *Greedy policy*: in the case of a finite-horizon problem and given a function $v : \{0, \ldots, N-1\} \times \mathcal{S} \to \mathbb{R}$, a policy is greedy (for v) if for each time n and state s we have

$$\pi_n(s) \in \operatorname{argmax}_a r(s, a) + \int_{s'} v(n+1, s') p(s'|s, a) ds',$$

[15] The readers accustomed with (stochastic) optimal control must note that what they usually call value function is called here optimal value function.

[16] We have of course $Q^\pi(n, s, a) = r(s, a) + \int_{s'} V^\pi(n+1, s') p(s'|s, a) ds'$ and $V^\pi(n, s) = Q^\pi(n, s, \pi(s))$.

[17] We have of course $Q^\pi(s, a) = r(s, a) + \gamma \int_{s'} V^\pi(s') p(s'|s, a) ds'$ and $V^\pi(s) = Q^\pi(s, \pi(s))$.

[18] We have of course (in the case of infinite-horizon problems) $Q^*(s, a) = r(s, a) + \gamma \int_{s'} V^*(s') p(s'|s, a) ds'$ and $V^*(s) = \max_a Q^*(s, a)$.

and the concept is similarly defined in an infinite-horizon problem for a function $v : \mathcal{S} \to \mathbb{R}$ by

$$\forall s \in \mathcal{S}, \quad \pi(s) \in \text{argmax}_a r(s,a) + \gamma \int_{s'} v(s') p(s'|s,a) ds'.$$

It is noteworthy that any policy that is greedy for V^* is an optimal policy because of the dynamic programming principle.

1.1.2.3 Main RL methods

In order to discuss the main methods of the RL literature, or at least the main ideas underlying them, it is useful to consider separately the two different objective functions considered above. We start with the case of infinite-horizon problems and go on with the case of finite-horizon ones.

Infinite-horizon problems

Many methods have been proposed for the case of infinite-horizon (stationary) problems.[19] In order to understand the main families of methods let us first introduce Bellman equations and the associated operators.

Bellman equations are functional equations solved by the value functions. For a given policy π, V^π is solution of the linear Bellman equation

$$V^\pi(s) = r(s, \pi(s)) + \gamma \int_{s'} p(s'|s, \pi(s)) V^\pi(s') ds',$$

that we can write with a linear operator as $V^\pi = \mathcal{T}^\pi V^\pi$. As far as the optimal value function is concerned, we have another Bellman equation, nonlinear in that case,

$$V^*(s) = \max_a r(s,a) + \gamma \int_{s'} p(s'|s,a) V^*(s') ds',$$

that we can write with a nonlinear operator as $V^* = \mathcal{T}^* V^*$. Similar equations exist for the functions Q^π and Q^*. The interesting point is that value functions are fixed points of contracting operators (because $\gamma \in (0,1)$).

Given the latter remark, it is natural to introduce a method – called Value Iteration – that starts from an initial function V_0 and iteratively builds a sequence

[19] It is always possible – although not recommended – to approximate an infinite-horizon problem by a finite-horizon problem by fixing an end date N sufficiently large. The methods of the next section could be used in that case. In all cases, an infinite-horizon problem can also be seen as a finite-horizon problem with a random terminal time (following a geometric distribution in the case of a constant discount rate γ as above).

$(V_k)_k$ by $V_{k+1} = \mathcal{T}^* V_k$.[20] This method converges[21] but it is often infeasible in practice when the state space and the action space are too large. Approximate dynamic programming techniques (which are part of RL techniques) can then help to beat the curse of dimensionality by replacing the above recursive equation by another of the form $V_{k+1} = \tilde{\mathcal{A}}\mathcal{T}^* V_k$ where $\tilde{\mathcal{A}}$ is an approximation operator. In practice this means that value functions are parametrized (they take the form of a neural network or a linear combination of well-chosen features) and that, at step k, values of $\mathcal{T}^* V_k$ are sampled for many points in order to feed a supervised learning algorithm (represented by $\tilde{\mathcal{A}}$) that allows to obtain V_{k+1} in a (similar) parametrized form. This type of approximate dynamic programming methods is interesting but the algorithms do not always converge.

Another family of methods is called Policy Iteration. It does not use the Bellman equation for the optimal value function but rather that for given policies. It starts from an initial (stationary) policy and works through iterations of policy evaluation and policy improvement steps. During policy evaluation, one evaluates the value function of the current policy, while at policy improvement steps, one updates the policy by considering a greedy policy with respect to the value function computed in the evaluation step (or at least moves the policy from the current one towards one that is closer to the greedy one).

In order to evaluate the value function associated with a policy, classical methods include solving the linear Bellman equation on a grid, the use of an iterative method as in the Value Iteration algorithm (with the operator $\mathcal{T}^\pi$ instead of $\mathcal{T}^*$), Monte-Carlo techniques, or the use of temporal differences (TD). The use of TD learning is inspired from stochastic approximation and is really one of the cornerstones of many RL algorithms. In particular, when using TD learning, one does not require to know the transition kernel and instead works with realized or simulated data.

The main idea behind TD learning, when applied to a value function V^π, is that one can build approximations thanks to a realization $(s_n, a_n, r_{n+1})_n$ of the MDP for a policy π by updating (in a synchronous or asynchronous manner) the current approximation $\hat{V}^\pi(s_n)$ of the value function at point s_n in the direction of $r_{n+1} + \gamma \hat{V}^\pi(s_{n+1}) - \hat{V}^\pi(s_n)$. There are of course many variants based on similar ideas. When tabular methods cannot be applied because of the size of the state space and when value functions are instead parametrized, TD learning methods update the parameters "so as" to reduce the gap between $r_{n+1} + \gamma \hat{V}^\pi(s_{n+1})$ and $\hat{V}^\pi(s_n)$ – see for instance (Sutton and Barto, 2018) and (Szepesvári, 2010) for a detailed introduction, in particular as semi-gradient methods are often used and

[20] By definition of the $\mathcal{T}^*$ operator, the use of Value Iteration requires information about the transition kernel, in particular the state transition kernel and the expected rewards.

[21] We mean convergence in terms of value functions. This does not mean that the associated sequence of greedy policies (which is often chosen) converges.

cannot be described in depth in this subchapter.

In practice, one usually does not wait for a precise evaluation of the value function associated with a policy before changing the policy: the value function, regarded as a "critic", evolves progressively to guide the "actor" in his policy changes. Also, many RL methods are based on a parametrized policy (with neural networks for instance) as is the case for the value function. In that case, the parameters of the policy are updated to move the policy in what we believe is the right direction according to the current value function.

Regarding the policy improvement step, it is noteworthy that determining the greedy policy associated with a value function requires to know the transition kernel. A classical alternative consists, instead of evaluating the value function V^π of the policy π in the policy evaluation step, in evaluating the state-action value function Q^π (using methods similar to those used for V^π). The policy improvement step then boils down to finding for each state s the maximum of $Q^\pi(s, \cdot)$ if one wants to be greedy.

There are in fact many methods based on Q functions. The application of TD learning ideas to Q functions leads to two important standard algorithms: SARSA and Q-learning (see the classical textbooks cited above for detailed presentations). These algorithms are aimed at directly finding the optimal state-action value function and exist in many variants, in particular when Q functions are parametrized with a neural network (hence the expression Q-network agent), and have known great successes in playing games.

Overall, it is noteworthy that the use of TD learning ideas enables to learn without knowledge of the underlying model. Combined with approximation techniques that enable to beat the curse of dimensionality, these ideas are central to modern RL techniques. Of course, the devil often lies in the details and it is clear that beyond the basic ideas presented in this subchapter, experience in the design of RL agents has strongly contributed to recent successes (e.g. the choice of learning rates in TD learning, the use of several neural networks, the use of experience replay, the architecture of neural networks, the parallelization of computations, the use of GPU and TPU, etc.).

Finite-horizon problems

The case of finite-horizon problems has to be considered separately from that of infinite-horizon problems although many ideas are common to both. In what follows we first briefly discuss value function methods and then go on with direct policy search methods.

As in the case of infinite-horizon problems, the value functions of finite-horizon

problems are characterized by Bellman equations. For a given policy π, V^π is indeed a solution to the linear Bellman equation

$$V^\pi(n,s) = r(s,\pi_n(s)) + \int_{s'} p(s'|s,\pi_n(s))V^\pi(n+1,s')ds',$$

i.e. $V^\pi(n,\cdot) = \mathcal{T}^\pi V^\pi(n+1,\cdot)$, for $n < N$, and $V^\pi(N,\cdot) = R(\cdot)$. As for the optimal value function, it solves the nonlinear Bellman equation

$$V^*(n,s) = \max_a r(s,a) + \int_{s'} p(s'|s,a)V^*(n+1,s')ds',$$

i.e. $V^*(n,\cdot) = \mathcal{T}^* V^*(n+1,\cdot)$, for $n < N$, and $V^*(N,\cdot) = R(\cdot)$. Similar equations exist for the functions Q^π and Q^* in this case.

Some of the methods presented in the previous section can be adapted to the finite-horizon case. For instance, methods inspired from Policy Iteration are well suited and work almost as if time was part of the state space. For the policy evaluation step, if tabular methods cannot be used because of the dimensionality of the problem, Monte Carlo and TD methods with approximation can be used. For the policy improvement step, it can be done date by date in an independent manner, except if one is looking for a policy in the form of a parametric function that depends on time, as it is sometimes the case.

In the case of finite-horizon problems, it is often interesting to use methods based on the backward induction underlying the dynamic programming principle. Indeed, the problem can be decomposed into N one-step problems, assuming for each date that we know how to solve the tail problem. In addition to the classical tabular method that consists in solving the Bellman equation for V^* step by step – backward in time – on a grid, there are classical approximate dynamic programming methods that approximate sequentially – backward in time – the functions $(V^*(n,\cdot))_n$ or $(Q^*(n,\cdot,\cdot))_n$ using regression techniques (see for instance (Bertsekas and Tsitsiklis, 1996) or the recent papers (Bachouch et al., 2018; Huré et al., 2018)). One of the main difficulties with these methods is that we do not know where to sample points at each step, because we do not know where we are going to need approximations of the value function in the future (that is, at previous time steps of the dynamic optimization problem).

In the case of finite-horizon problems, it is possible to use RL methods that do not rely on value functions, but instead directly search for the optimal policy – hence their name: direct policy search methods. In direct policy search methods, policies π^θ are parametrized by a vector θ (often the coefficients of a neural network or those of a simple linear combination of well-chosen features[22]) and the goal is to maximize over θ the expected score $\mathbb{E}\left[\sum_{n=0}^{N-1} r(S_n,\pi^\theta(S_n)) + R(S_N)\right]$.

[22]The coordinates of θ can also be the values of the policy if the state space is small.

The problem becomes therefore a pure stochastic optimization problem and many approaches are available, going from gradient methods to non-gradient methods (simulated annealing, evolutionary approaches, etc.). It is noteworthy that it is often interesting to use stochastic policies in this context (see for instance the REINFORCE algorithm and the numerous variants that have been proposed to accelerate convergence).

Direct policy search methods are interesting but they often suffer from the vanishing/exploding gradient phenomenon because of the recurrent nature of the problem (a change of action at date k potentially changes indeed all states and rewards after date k). To avoid this, it is sometimes possible to proceed by backward induction: approximate the optimal decision rules for the last periods using a direct policy search approach and freeze them to approximate the optimal decision rules for the previous periods, and so on (see for instance (Bachouch et al., 2018; Huré et al., 2018)). In that case, the same sampling problem as above occurs once again.

The above ideas are general and have to be adapted to each context of application. Let us now discuss what makes finance specific when it comes to the use of RL ideas.

1.1.3 Finance is not a game

The craze around RL-based computer programs triggered by the successes of DeepMind goes along, in the financial community, with the hope that the progress made in board and video games can be translated into progress for algorithmic trading. However, expectations need to be managed when it comes to finance problems. Although many finance problems are dynamic optimization problems that could be addressed with the tools of RL, they do not have the same characteristics as the toy problems (Cart Pole, Mountain Car, etc.) of the RL community and are different from those recently addressed with success.

1.1.3.1 States and actions

A first important difference between games or toy problems and finance problems has to do with the definition of the state space. In the former, the state space may be complicated but it is naturally and unambiguously defined because it is imposed by the problem: the place of pieces on the board in the case of chess or draughts, the pixels (with history maybe) in the case of video games, etc. In the case of a finance problem, the state space is open and must definitely be regarded as a modelling choice. In the case of problems involving a limit order book (LOB) for instance, the state of the LOB (which can usually be reduced to a few limits) – and maybe its history – is naturally part of the story but the state space can also include numerous signals related to trends, historical or implied volatilities,

market volumes, etc. There is in fact no limit to the size of the state space and one does not know a priori amongst the numerous candidate variables those that are relevant and those that should be discarded.

Regarding the action space, it can be discrete or continuous, sometimes partly discrete and partly continuous (think of the discrete choice – because of the tick size – of the limit price and the continuous – although in practice it involves integers – choice of the order size, in order placement problems), but it is usually easily defined in algorithmic trading problems.

1.1.3.2 The role of models

One could naively think that, as many RL methods do not require the knowledge of the transition kernel, the latter does not really matter. In fact, the recent successful RL agents have been trained on simulated data. As a consequence the transition kernel was "known" by the simulator, even though it was not used by the learning algorithm. In particular, it is essential to understand that the often-employed expression "model-free" only applies to the learning method. That said, is a model-based simulator essential for addressing algorithmic trading problems, or, more generally, finance problems, with RL tools? As we shall see, the answer is often positive[23] and this has important consequences on what we can expect from RL methods in algorithmic trading.

Many finance problems can be solved using the tools of optimal control. When it is not the case, RL methods need samples. A tempting alternative to a model-based simulator could be the use of historical data as a way to sample data. This is often proposed but raises a lot of concerns. First, historical data is typically not sufficient if one wants to take account of microstructural concerns (for instance priority issues in LOB cannot be addressed with common high-frequency data sets) or essential feedback effects like market impact. Second, many RL algorithms fail when used on a pre-collected batch of data as is the case when one uses historical data (see for instance (Fujimoto et al., 2019)). Third, historical data is scarce. In the case of a single-asset trading algorithm, training it on high-frequency LOB data may work depending on the context, but when it comes to multi-asset frameworks or lower-frequency data, most RL methods require far more data points than what historical datasets can offer. In fact, the usual rule of thumb stating that the number of data points should exceed by far the number of parameters to estimate applies to RL techniques and often disqualifies the raw use of historical data.[24]

[23] The online use of bandit algorithms to choose among a list of algorithms to carry out a given task is of course an exception, by nature.

[24] This remark is particularly important when it comes to multi-asset portfolio construction.

Among the consequences of the above discussion, one is particularly important to highlight: most RL methods (even the so-called "model-free" methods) can only solve a finance problem given a choice of state space and a model of kernel transition. Given that financial market dynamics do not follow specified rules and are non-stationary – unlike what happens with games or other problems – the solution found through RL methods can only be useful as long as the market, as a system, behaves as described by the model / the simulated data. Expecting too much from RL in algorithmic finance is like expecting a RL-based agent to play a new game every day when it has only been trained to play a given list of games.

The craze around RL should not lure financial practitioners. There is indeed no magic: non-stationarity will not disappear thanks to RL techniques. Model-free RL methods are nevertheless very useful in that they can be used upon simulated data based on different models without knowing these models. In other words, the place of models might be reduced in the future to data generation/simulation, should RL techniques be adopted.[25] [26]

1.1.3.3 The question of risk

In the RL literature, objective functions are almost always expected rewards. In finance however, maximizing expectation is often not enough as risk must be mitigated. This seems to disqualify RL, but this is of course not the case. In fact, in the design of MDP for financial applications, risk must be embedded in rewards.[27]

A classical approach in algorithmic trading, where the basic measure of performance is Profit & Loss (PnL), consists in maximizing a final reward that accounts for the risk. Maximizing a Von Neumann-Morgenstern expected utility of the PnL is an important example. Maximizing a risk-adjusted performance indicator such as a final Sharpe ratio is another. Differential versions of these approaches allow to go from a final reward to running rewards, which may be preferable for some learning algorithms.

Another typical (brute force) approach consists in using penalty terms in running rewards in order to penalize for risk at each date. This is for instance a common approach in market making models where large inventories are penalized by a local variance term.

[25] Even with good-quality simulators, learning will not be easy. Financial data is indeed often characterized by a low signal-to-noise ratio (compared to toy examples for instance). The only good point with low signal-to-noise ratio data is that exploration is somehow carried out naturally on some financial state variables.

[26] Some market multi-agent-based simulators are based on RL ideas (see for instance (Lussange et al., 2019a,b)).

[27] A classical approach to transform mean-variance dynamic optimization problems into classical time-consistent problems is that of (Zhou and Li, 2000).

1.1.3.4 The question of time steps

We previously discussed the construction of the MDP. Other important points could be highlighted when it comes to finance problems.

For example, in most problems, decisions are taken sequentially and the usual MDP setup is well adapted. The question of time may however be important for some specific problems in which the agent only (re)acts upon the occurrence of some specified events while the system evolves on its own between events. In that case, MDP may not be enough and more general frameworks are needed.

1.1.4 A review of existing works

As discussed above, RL techniques have been proposed for the pricing and hedging of contingent claims. However, option pricing is clearly out of the scope of this subchapter on algorithmic trading. Even though portfolio decisions are more and more made algorithmically and at higher and higher frequency and despite numerous research works suggesting the use of RL for optimizing portfolios, we do not cover optimal portfolio choice either. One reason is that this subfield of quantitative finance is traditionally not considered part of the algorithmic trading field. More importantly, another reason is that, in spite of their initial appeal,[28] most approaches proposed in the literature are dubious as they use structures (for instance neural networks) in which the number of degrees of freedom (i.e. the number of parameters) is large given the volume of historical data used for training (in particular when low-frequency data is used and when the number of assets is large).[29] [30]

Let us now review existing works involving RL techniques for addressing three kinds of problems: the design of statistical arbitrage strategies,[31] the optimization of execution strategies, and the optimization of market making strategies.

1.1.4.1 Statistical arbitrage

The literature on statistical arbitrage is very diverse. Overall, papers in this field have often been written to advocate for the use of particular tools and techniques,

[28] Merging the estimation/forecasting step with the investment decision step is indeed attractive.

[29] The use of synthetic market data is sometimes proposed to circumvent this (overfitting) problem but the literature is, as of today, limited – see for instance (Wiese et al., 2020) or (Yu et al., 2019).

[30] Interesting ideas have recently been developed to avoid the above problems while using RL techniques. See for instance (Wang, 2019) and (Wang and Zhou, 2020).

[31] The design of statistical arbitrage strategies can be regarded as some form of portfolio choice problem. However, for one or two-asset problems with high-frequency data, RL techniques may be less prone to overfitting.

or at least to exemplify them.[32] RL tools appeared in the literature on statistical arbitrage more than twenty years ago. The first wave of papers was followed by another one in recent years to advocate again for the use of RL techniques. Our goal here is to shed light on a few representative papers to help the readers in their own research in the field.[33]

In a series of papers including (Moody et al., 1998) and (Moody and Saffell, 2001) – see also the references therein – researchers proposed direct policy search methods to optimize the strategy of a trader. In particular, in (Moody and Saffell, 2001), the authors built a mid-frequency long/short trading strategy on USD/GBP based on 30-minute data. Their method[34] used a neural network for the position to take with a short history of past returns as inputs, and a gradient ascent in order to optimize various risk-adjusted performance measures such as differential forms of Sharpe and Sortino ratios, while taking into account transaction costs. Interestingly, they found little evidence for the need of exploration, probably because the signal-to-noise ratio is so small in finance that it naturally induces exploration. (Gold, 2003) used a very similar approach to design a trading strategy on several currency pairs. An extension of the approach with a risk management layer and a dynamic optimization layer was proposed in (Dempster and Leemans, 2006) with 1-minute EUR/USD market data.

Recently, a similar approach was proposed in (Lu, 2017) with the additional use of a LSTM network. Approaches based on value functions have also been proposed in the second wave of papers. (Cumming et al., 2015) presented a policy iteration approach in order to maximize the expected PnL of a high-to-mid frequency foreign exchange (FX) trader. More precisely, they used 1-minute candlesticks history on several currency pairs to train an algorithm working through LSTD (a classic of TD learning algorithms) for policy evaluation and improving policies with a standard greedy step. (Carapuço et al., 2018) proposed more recently a FX trading algorithm based on a deep Q-network (DQN) trained on a dataset that accounts for market microstructure, but only mid-frequency trading decisions are considered.

Most of the papers using RL tools to design statistical arbitrage strategies are applied to FX markets. Outside of FX markets, we can first cite (Deng et al., 2016). Their approach is based on a gradient-based direct policy search where strategies are represented through a neural network. They exemplified their algorithm on 1-minute data for a Chinese stock index futures and two commodity futures.[35] Recently, (Théate and Ernst, 2020) also proposed a MDP framework

[32]People building strategies that work in real financial markets are indeed unlikely to publish papers with full details.

[33]Our goal is not to be exhaustive. Also, we do not really assess the quality of the papers.

[34]They also tested a Q-learning approach and obtained results in favour of direct policy search.

[35]They obtained better results with another approach that is not based on RL in (Deng et al.,

for optimizing (with a DQN)[36] algorithmic trading strategies. Their framework is very rich and detailed with open-high-low-close data, macroeconomic indicators, and even news as state variables, together with realistic rewards involving transaction costs, but their application is restricted to simple contexts with daily data. The same researchers, however, along with co-authors, proposed in (Boukas et al., 2020) a RL approach for intraday bidding on energy markets.

1.1.4.2 Optimal execution

The optimal execution of orders raises a lot of issues associated with the trade-off between liquidity costs and volatility, but also with deep market microstructural questions. The literature on optimal execution started around two decades ago with the seminal papers (Almgren and Chriss, 1999, 2001) tackling the optimal scheduling problem of agents willing to balance, on the one hand, their incentive to trade fast in order to avoid market fluctuations, and, on the other hand, their incentive to trade slowly in order to have as little impact and liquidity-related costs as possible. Since then, many research works have been carried out on the optimal scheduling problem with different modelling assumptions regarding market impact and transaction costs, and different objective functions. Beyond the optimal scheduling problem that focuses on the splitting of a large parent order into child orders, another strand of research has focused on the child order placement problem.[37] Some papers focus on the trade-off between (i) posting a liquidity-providing limit order and having no guarantee of execution but a good execution price, and (ii) sending liquidity-taking orders but paying the bid-ask spread. Some others study the optimal routing of orders in a fragmented market with many lit and dark pools.[38]

Most of the papers in the literature are based on optimal control tools. RL techniques could therefore be a way to consider models with more state variables and more complex dynamics.

One of the first papers to advocate for the use of RL techniques to solve optimal execution problems is (Nevmyvaka et al., 2006). The problem addressed in that paper is that of an agent willing to sell a given number of shares within a short period of time (a few minutes in their case) by posting a single limit

2015).

[36]The possibility to use Q-learning to build statistical arbitrage strategies is also exemplified in (Ritter, 2017). In a very simple simulation model where returns are mean reverting, the author built a Q-learning agent that takes into account trading costs, market impact, and a form of risk aversion through a quadratic penalty in rewards.

[37]The former problem corresponds to the strategic layer of most execution algorithms while the latter corresponds to their tactical layer.

[38]Optimal execution has been one of the very active fields of quantitative finance in the last decade. We refer to the books (Cartea et al., 2015) and (Guéant, 2016) for a detailed description of the field.

order in a LOB – and potentially updating it – or crossing the spread. Their approach is a brute force form of Q-learning in tabular mode trained on NASDAQ high-frequency data. Although it is simplistic with a low-dimensional state space suffering from too much discretization, that paper paved the way to other RL applications to optimal execution. It surprisingly remains, however, one of the only RL papers dealing with limit order placement. An exception is the recent paper (Schnaubelt, 2020) that deals with limit order placement in cryptocurrency markets. Another exception is the very recent paper (Karpe et al., 2020) that uses the market simulator of (Byrd et al., 2019) to build a double DQN agent that places limit orders or market orders as a function of remaining quantity and time, but also bid-ask spread, market imbalance, and past evolution of prices.

Regarding optimal scheduling,[39] (Hendricks and Wilcox, 2014) used Q-learning to perform better than the trading curves of Almgren-Chriss models, by making decisions not only based on remaining inventory and time, but also on bid-ask spreads and volumes. The simple model they used relied on discretized variables so as to be able to use tabular methods, and learnt on 5-minute bins constructed with granular historical data for South African stocks (ignoring therefore the impact of actions on the market variables). (Ning et al., 2018) considered a double DQN approach to train an agent that makes decisions (based on remaining inventory, time, mid-price, and a volatility measure) on the size of the next market order in an Almgren-Chriss-like framework with no information on the price dynamics. Their training set was based on 1-second mid-price historical data and therefore, once again, no feedback of the actions on the market dynamics was taken into account.[40] Their approach probably constitutes one of the most interesting starting points for real RL-based execution algorithms. (Dabérius et al., 2019) is another interesting paper that compared the use of a double DQN and that of a policy-based approach for solving problems inspired from the optimal execution models of (Cartea et al., 2015). Strangely, they did not test their results on historical data or simulated data backed by historical data.

In the optimal execution field, dark pool exploration has also been addressed using online RL tools. We refer to (Ganchev et al., 2010) and (Laruelle et al., 2011) and to the subchapter written by Sophie Laruelle for more details.

[39] Optimal execution models that only consider market orders should be regarded as optimal scheduling models.

[40] They argue that historical data is enough to train the model if it is then continuously updated when used in reality. This is an interesting idea, but it requires an intensive usage of the algorithm to be able to account for market impact.

1.1.4.3 Market making

Economists interested in market microstructure have studied the behaviour of market makers / dealers / market specialists for a long time with the aim of understanding market liquidity and the different factors explaining the very existence or the magnitude of bid-ask spreads. The two usual types of models are (i) models where one or several risk-averse market makers optimize their pricing policy for managing their inventory risk models (see (Amihud and Mendelson, 1980), (Ho and Stoll, 1980, 1981, 1983), and (O'Hara and Oldfield, 1986)) and (ii) models focused on information asymmetries where bid-ask spreads derive from adverse selection (see for instance (Copeland and Galai, 1983) or (Glosten and Milgrom, 1985)). Other classic economic references on market making include (Grossman and Miller, 1988) and the review paper (Stoll, 2003).

In 2008, largely inspired by (Ho and Stoll, 1981), (Avellaneda and Stoikov, 2008) proposed a stochastic optimal control model to determine the optimal bid and ask quotes that a single-asset risk-averse market maker should set. The authors paved the way to a new literature on market making that put more focus on the very problem faced by a market maker; unlike the previous one that rather focused on a general understanding of liquidity. The models of this new literature can be divided into two groups: those adapted to the problem of a market maker in a limit order book and those adapted to OTC markets where market making automation is now commonplace (bonds, FX, etc.). Most of them use stochastic optimal control tools (see the books (Cartea et al., 2015) and (Guéant, 2016) for detailed discussions) but many RL approaches have been proposed over the recent past.

The use of RL techniques for market making automation is, however, not a new idea. The earliest paper is indeed (Chan and Shelton, 2001). The goal of the authors was clearly to advocate for the use of RL techniques: they proposed a model inspired from (Glosten and Milgrom, 1985) for the market with informed and uninformed traders, and used several RL methods (Monte-Carlo, SARSA, actor-critic) in order to find the optimal quotes of their market maker. They allow for the use of several relevant state variables (inventory, imbalance, market quality) and several forms of rewards including proxies of the risk borne by the market maker. Of course their model was too simple for any practical use but they clearly anticipated the relevance of RL tools in the field of automated market making.[41]

In recent years, the renewed popularity of RL techniques has been associated with a significant number of new RL papers dealing with market making.

[41] Their work in a Glosten-Milgron information model inspired the recent paper (Mani et al., 2019).

For order-driven markets, the most cited reference is certainly (Spooner et al., 2018). The authors of that article used historical LOB data to train a market maker using several RL methods (SARSA, Q-learning, double Q-learning, etc.) in a model where the state space is simplified thanks to tile coding approximation. That paper is interesting as it constitutes a good starting point for future research. Furthermore, it sheds light on the need for a market simulator: the authors indeed acknowledged that priority issues in LOB cannot be addressed by only using common historical LOB data.

For quote-driven markets, (Guéant and Manziuk, 2019) proposed several actor-critic approaches in which the value function and the policies are approximated with neural networks. Using RL techniques, they showed how to approximate the optimal quotes in one of the multi-asset market making models proposed in (Guéant, 2017). Their applications concerned a portfolio of 20 corporate bonds.[42] Another interesting paper for OTC markets is (Ganesh et al., 2019) in which the authors train a market maker thanks to a policy-search approach in a simulated market with several dealers.

Other recent works include (Lim and Gorse, 2018) and (Kumar, 2020) but the version of the papers we had access to did not contain enough details.

1.1.5 Conclusion and perspectives for the future

In finance, many problems can be modelled with MDP: portfolio choice, hedging in complete and incomplete markets, optimal execution, market making, etc. This mathematical framework being exactly that of RL, the enthusiasm around RL techniques following the recent successes of DeepMind came with the hope of being able to solve most of the problems usually addressed using MDP and / or the tools of (stochastic) optimal control. In particular, academics and quantitative analysts in the financial industry hoped to get alternatives to numerical methods based on grids – which are known to suffer from the curse of dimensionality – in order to solve high-dimensional problems. Another hope was to get rid of the simplifying assumptions on the dynamics of financial variables in most models. RL indeed came with the promise that models were not always necessary to address dynamic optimization problems, i.e. that observations could be enough (a claim that we clarified above for financial applications).

In this short paper, we have put into perspective the use of RL techniques for addressing finance problems. In addition to highlighting what made finance special compared to games and other fields where RL led to successes, we have

[42] (Baldacci et al., 2019) also used an actor-critic approach to solve a two-stage principal-agent problem involving an exchange (which sets fees) and a market maker.

insisted on the need to carry out important works in order to obtain satisfactory market simulators – with characteristics that depend on the problem type – for training RL algorithms.

We have also presented examples of articles using RL techniques to build simple statistical arbitrage trading algorithms or to solve optimal execution and market making problems. These articles should be regarded as proofs of concept and we believe it is now time for building and using scalable RL-based execution and market making trading algorithms within financial institutions.

As noted by several renowned scientists, the recent breakthroughs involving RL are mainly technological, not scientific. For instance, Dimitri Bertsekas, one of the greatest specialists of optimal control, claimed that the great success of AlphaZero was due to a "skillful implementation/integration of known ideas, and awesome computational power". Subsequently, a necessary condition for soon seeing RL-based trading agents in many financial institutions is that traditional quants, computer scientists, and engineers unite forces and ride the learning curve together.

Bibliography

Abergel, F., Huré, C., and Pham, H. (2020). Algorithmic trading in a microstructural limit order book model. *Quantitative Finance*, pages 1–21.

Almgren, R. and Chriss, N. (1999). Value under liquidation. *Risk*, 12(12):61–63.

Almgren, R. and Chriss, N. (2001). Optimal execution of portfolio transactions. *Journal of Risk*, 3:5–40.

Amihud, Y. and Mendelson, H. (1980). Dealership market: Market-making with inventory. *Journal of financial economics*, 8(1):31–53.

Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Avellaneda, M. and Stoikov, S. (2008). High-frequency trading in a limit order book. *Quantitative Finance*, 8(3):217–224.

Bachouch, A., Huré, C., Langrené, N., and Pham, H. (2018). Deep neural networks algorithms for stochastic control problems on finite horizon, part 2: Numerical applications. *arXiv preprint arXiv:1812.05916*.

Baldacci, B., Manziuk, I., Mastrolia, T., and 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*.

Bäuerle, N. and Rieder, U. (2011). *Markov decision processes with applications to finance*. Springer Science & Business Media.

Becker, S., Cheridito, P., and Jentzen, A. (2019). Deep optimal stopping. *Journal of Machine Learning Research*, 20:74.

Bertsekas, D. P. (2019). *Reinforcement learning and optimal control*. Athena Scientific Belmont, MA.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*.

Boukas, I., Ernst, D., Théate, T., Bolland, A., Huynen, A., Buchwald, M., Wynants, C., and Cornélusse, B. (2020). A deep reinforcement learning framework for continuous intraday market bidding. *arXiv preprint arXiv:2004.05940*.

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

Byrd, D., Hybinette, M., and Balch, T. H. (2019). Abides: Towards high-fidelity market simulation for ai research. *arXiv preprint arXiv:1904.12066*.

Carapuço, J., Neves, R., and Horta, N. (2018). Reinforcement learning applied to forex trading. *Applied Soft Computing*, 73:783–794.

Cartea, Á., Jaimungal, S., and Penalva, J. (2015). *Algorithmic and high-frequency trading*. Cambridge University Press.

Chan, N. T. and Shelton, C. (2001). An electronic market-maker.

Charpentier, A., Elie, R., and Remlinger, C. (2020). Reinforcement learning in economics and finance. *arXiv preprint arXiv:2003.10014*.

Copeland, T. E. and Galai, D. (1983). Information effects on the bid-ask spread. *the Journal of Finance*, 38(5):1457–1469.

Cumming, J., Alrajeh, D., and Dickens, L. (2015). An investigation into the use of reinforcement learning techniques within the algorithmic trading domain. *Imperial College London: London, UK*.

Dabérius, K., Granat, E., and Karlsson, P. (2019). Deep execution-value and policy based reinforcement learning for trading and beating market benchmarks. *Available at SSRN 3374766*.

Dempster, M. A. and Leemans, V. (2006). An automated fx trading system using adaptive reinforcement learning. *Expert Systems with Applications*, 30(3):543–552.

Deng, Y., Bao, F., Kong, Y., Ren, Z., and 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.

Deng, Y., Kong, Y., Bao, F., and Dai, Q. (2015). Sparse coding-inspired optimal trading system for hft industry. *IEEE Transactions on Industrial Informatics*, 11(2):467–475.

E, W., Han, J., and 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.

Fischer, T. G. (2018). Reinforcement learning in financial markets-a survey. Technical report, FAU Discussion Papers in Economics.

Fujimoto, S., Meger, D., and Precup, D. (2019). Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR.

Ganchev, K., Nevmyvaka, Y., Kearns, M., and Vaughan, J. W. (2010). Censored exploration and the dark pool problem. *Communications of the ACM*, 53(5):99–107.

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

Glosten, L. R. and Milgrom, P. R. (1985). Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of financial economics*, 14(1):71–100.

Gold, C. (2003). Fx trading via recurrent reinforcement learning. In *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003. Proceedings.*, pages 363–370. IEEE.

Grossman, S. J. and Miller, M. H. (1988). Liquidity and market structure. *the Journal of Finance*, 43(3):617–633.

Guéant, O. (2016). *The Financial Mathematics of Market Liquidity: From optimal execution to market making*, volume 33. CRC Press.

Guéant, O. (2017). Optimal market making. *Applied Mathematical Finance*, 24(2):112–154.

Guéant, O. and Manziuk, I. (2019). Deep reinforcement learning for market making in corporate bonds: beating the curse of dimensionality. *Applied Mathematical Finance*, 26(5):387–452.

Han, J., Jentzen, A., and E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510.

Hendricks, D. and Wilcox, D. (2014). A reinforcement learning extension to the almgren-chriss framework for optimal trade execution. In *2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, pages 457–464. IEEE.

Henry-Labordere, P. (2017). Deep primal-dual algorithm for bsdes: Applications of machine learning to cva and im. *Available at SSRN 3071506*.

Ho, T. and Stoll, H. R. (1980). On dealer markets under competition. *The Journal of Finance*, 35(2):259–267.

Ho, T. and Stoll, H. R. (1981). Optimal dealer pricing under transactions and return uncertainty. *Journal of Financial economics*, 9(1):47–73.

Ho, T. S. and Stoll, H. R. (1983). The dynamics of dealer markets under competition. *The Journal of finance*, 38(4):1053–1074.

Huré, C., Pham, H., Bachouch, A., and Langrené, N. (2018). Deep neural networks algorithms for stochastic control problems on finite horizon, part i: convergence analysis. *arXiv preprint arXiv:1812.04300*.

Huré, C., Pham, H., and Warin, X. (2019). Some machine learning schemes for high-dimensional nonlinear pdes. *arXiv preprint arXiv:1902.01599*.

Karpe, M., Fang, J., Ma, Z., and Wang, C. (2020). Multi-agent reinforcement learning in a realistic limit order book market simulation. *arXiv preprint arXiv:2006.05574*.

Kolm, P. N. and Ritter, G. (2020). Modern perspectives on reinforcement learning in finance. *Modern Perspectives on Reinforcement Learning in Finance (September 6, 2019). The Journal of Machine Learning in Finance*, 1(1).

Kumar, P. (2020). Deep reinforcement learning for market making. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1892–1894.

Laruelle, S., Lehalle, C.-A., and Pages, G. (2011). Optimal split of orders across liquidity pools: a stochastic algorithm approach. *SIAM Journal on Financial Mathematics*, 2(1):1042–1076.

Lim, Y.-S. and Gorse, D. (2018). Reinforcement learning for high-frequency market making. In *ESANN*.

Lu, D. W. (2017). Agent inspired trading using recurrent reinforcement learning and lstm neural networks. *arXiv preprint arXiv:1707.07338*.

Lussange, J., Bourgeois-Gironde, S., Palminteri, S., and Gutkin, B. (2019a). Stock price formation: useful insights from a multi-agent reinforcement learning model. *arXiv preprint arXiv:1910.05137*.

Lussange, J., Lazarevich, I., Bourgeois-Gironde, S., Palminteri, S., Gutkin, B., et al. (2019b). Stock market microstructure inference via multi-agent reinforcement learning. Technical report.

Mani, M., Phelps, S., and Parsons, S. (2019). Applications of reinforcement learning in automated market-making.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Moody, J. and Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, 12(4):875–889.

Moody, J., Wu, L., Liao, Y., and Saffell, M. (1998). Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, 17(5-6):441–470.

Munos, R. (2004). *Contributions à l'apprentissage par renforcement et au contrôle optimal avec approximation*. PhD thesis.

Nevmyvaka, Y., Feng, Y., and Kearns, M. (2006). Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, pages 673–680.

Ning, B., Ling, F. H. T., and Jaimungal, S. (2018). Double deep q-learning for optimal execution. *arXiv preprint arXiv:1812.06600*.

Noonan, L. (2017). Jpmorgan develops robot to execute trades. *Financial Times*.

O'Hara, M. and Oldfield, G. S. (1986). The microeconomics of market making. *Journal of Financial and Quantitative analysis*, pages 361–376.

Pagès, G., Pham, H., and Printems, J. (2004). Optimal quantization methods and applications to numerical problems in finance. In *Handbook of computational and numerical methods in finance*, pages 253–297. Springer.

Pham, H., Warin, X., and Germain, M. (2019). Neural networks-based backward scheme for fully nonlinear pdes. *arXiv preprint arXiv:1908.00412*.

Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons.

Ritter, G. (2017). Machine learning for trading. *Available at SSRN 3015609*.

Rockafellar, R. T. and Uryasev, S. (2002). Conditional value-at-risk for general loss distributions. *Journal of banking & finance*, 26(7):1443–1471.

Ruf, J. and Wang, W. (2020). Neural networks for option pricing and hedging: a literature review. *Journal of Computational Finance, Forthcoming*.

Schnaubelt, M. (2020). Deep reinforcement learning for the optimal placement of cryptocurrency limit orders. Technical report, FAU Discussion Papers in Economics.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

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

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Spooner, T., Fearnley, J., Savani, R., and Koukorinis, A. (2018). Market making via reinforcement learning. *arXiv preprint arXiv:1804.04216*.

Stoll, H. R. (2003). Market microstructure. In *Handbook of the Economics of Finance*, volume 1, pages 553–604. Elsevier.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103.

Théate, T. and Ernst, D. (2020). An application of deep reinforcement learning to algorithmic trading. *arXiv preprint arXiv:2004.06627*.

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.

Wang, H. (2019). Large scale continuous-time mean-variance portfolio allocation via reinforcement learning. *Available at SSRN 3428125*.

Wang, H. and Zhou, X. Y. (2020). Continuous-time mean–variance portfolio selection: A reinforcement learning framework. *Mathematical Finance*, 30(4):1273–1308.

Wiese, M., Knobloch, R., Korn, R., and Kretschmer, P. (2020). Quant gans: Deep generation of financial time series. *Quantitative Finance*, pages 1–22.

Yu, P., Lee, J. S., Kulyatin, I., Shi, Z., and Dasgupta, S. (2019). Model-based deep reinforcement learning for dynamic portfolio optimization. *arXiv preprint arXiv:1901.08740*.

Zhou, X. Y. and Li, D. (2000). Continuous-time mean-variance portfolio selection: A stochastic lq framework. *Applied Mathematics and Optimization*, 42(1):19–33.