Finance and Stochastics (2021) 26:103–129
https://doi.org/10.1007/s00780-021-00467-2

Reinforcement learning and stochastic optimisation

Sebastian Jaimungal[1]

Received: 31 January 2021 / Accepted: 19 October 2021

Abstract
At the heart of financial mathematics lie stochastic optimisation problems. Traditional approaches to solving such problems, while applicable to broad classes of models, require specifying a model to complete the analysis and obtain implementable results. Even then, the curse of dimensionality challenges the viability of conventional methods to settings of practical relevance. In contrast, machine learning, and reinforcement learning (RL) particularly, promises to learn from data and overcome the curse of dimensionality simultaneously. This article touches on several approaches in the extant literature that are well positioned to merge our traditional techniques with RL.

Keywords Stochastic optimisation · Stochastic games · Reinforcement learning · Machine learning

Mathematics Subject Classification (2020) 93E20 · 93E35 · 91G60 · 91G80

JEL Classification C02 · G11 · G12

1 Introduction

There are many interesting and exciting research directions around us, including (but not limited to) optimal transport, algorithmic trading and market microstructure, mean-field games, credit and systemic risk, model robustness, portfolio optimisation, and of course machine learning (ML). While I am keenly interested in many of these topics, I had to pick one for this article, and in my opinion, ML is touching all aspects of financial mathematics. The outlook that I provide here is far from complete (and

The author would like to acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) under the funding reference numbers RGPIN-2018-05705 and RGPAS-2018-522715.

✉ S. Jaimungal
sebastian.jaimungal@utoronto.ca

[1] Department of Statistical Sciences, University of Toronto, 700 University Avenue, Toronto, Ontario, Canada

I apologise for any omissions that I may have made), nor is it a "call to arms" to drop whatever it is you are doing and start working on ML-related problems. Instead, it is a biased view of what I deem are several exciting developments in ML that I suspect will and can significantly impact our craft both within academia and industry.

I have a love/hate relationship with ML – from one angle, there are numerous success stories in using ML techniques to solve problems ranging over portfolio optimisation, implied volatility surface modelling, algorithmic trading, mean-field games, and robo-advising, among many others. Nevertheless, when trying to glean the essential insights from these "successes", it is difficult to ascertain what new insights stem from ML applications in our field. A proponent may look at the empirical evidence of published works and preprints and say "We can now solve problems A, B and C,… using ML methodology, and we could not before!" A critic would say "Sure, we have some numerical results that work on particular examples of problems A, B and C,…, but what, other than the specific solution to this particular problem, have I learnt? What is the general structure of the solution? How may I interpret this result? How robust is the result of the particular ML model architecture?" I would argue that both views are correct. There are many successes, and many more will come along in surprising areas with impactful results from which our community may benefit. However, there is a gnawing feeling that often in these lines of work, we create tools to solve specific problems that, while relevant, do not provide fundamental insight. An important takeaway, however, from ML approaches is that, in the words of Tukey [113]: *"Far better an approximate answer to the right question, which is often vague, than an exact answer to the wrong question, which can always be made precise."*

ML methods and tools are vast and varied; thus, I must make some judicious choices on what to cover. As stochastic optimisation is a cornerstone of almost all aspects of financial mathematics and underpins much of our craft, I focus here on reinforcement learning (RL), which aims to solve variants of stochastic optimisation problems without making strong assumptions on the underlying stochastic dynamics or how the environment responds to agents' actions. I explore several methods and provide a few ideas for combining ML techniques with more traditional stochastic modelling techniques. I do not, however, proclaim to have the answers on how any of the ideas can be made precise. I do, nonetheless, hope that the exposition serves as a motivation for posing new problem formulations and acts as a catalyst for ideas of blending ML and financial mathematics, in contrast to merely borrowing ideas from ML to solve specific financial mathematics problems.

The remainder of this article starts with an overview of several deep learning methods for RL, and the subsequent sections touch on specific problems and discuss some potential ideas for blending ML techniques with stochastic analysis tools. The reader should be aware that the article is written with the lens that approximately solving problems that more accurately reflect data is an endeavour worth taking on. The article is neither rigorous nor complete; instead, the hope is to introduce ideas from the ML world that we, as financial mathematicians, should be made aware of and which hopefully spark insights into coupling (and formalising) ML techniques and ideas with those more familiar to us.

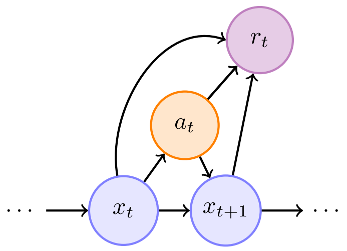


Fig. 1 Directed graph representation of the state–action–reward evolution

2 Deep reinforcement learning

This section provides an overview of deep RL approaches that I believe we as financial mathematicians should be aware of, and hopefully improve upon by blending these ideas with our traditional approaches.

In the simplest RL setting, the evolution of the environment, action and reward may be viewed as in the directed graph shown in Fig. 1. The system evolves (stochastically) from state x_t into state x_{t+1}, but that transition is affected by the agent's actions a_t, which itself depend on the state x_t, and the triplet (x_t, x_{t+1}, a_t) affects the agent's reward $r_t = r(x_t, a_t)$. We assume the dynamics are Markov (and stationary) in the state variables for ease of notation and let $\mathcal{A} \subseteq \mathbb{R}^m$ denote the set of allowable actions and $\mathcal{X} \subseteq \mathbb{R}^n$ the set of possible states.

The agent's goal is to determine the mapping from states to actions (called the *policy*) that maximises (discounted) total rewards or average running sum of total rewards. Thus, the agent's goal is to solve a stochastic control problem in discrete time, and when the state evolution is Markov, it is known as a Markov decision process (MDP); see Puterman [98, Chap. 4]. Numerous questions in financial mathematics fall within this class of problems ranging from portfolio optimisation to algorithmic trading to hedging.

In standard stochastic control, however, the stochastic evolution, the effect of actions and the rewards are all pre-specified and known. On the other hand, many RL approaches aim to perform the optimisation *online* (while making observations) and do so with little, or no strong, assumptions on the dynamics of the environment or the response the environment has from the agent's actions. Our conventional tools cannot handle such a "model-free" approach that learns from data. Even with specific models, traditional tools can solve only limited cases in closed form, and if the state dimension is above 3, our standard numerical tools face serious challenges. RL promises to tackle these difficulties simultaneously and head-on.

To describe the RL approach, we denote the total discounted reward at time t by $R_t^\pi = \sum_{s=t}^{\infty} \gamma^{s-t} r_t^\pi$, where $\gamma \in (0,1)$ is the discount factor, r_t^π is the random reward at time t under policy π, a mapping from states to actions, and the superscripts are present to remind the reader that the state path, actions and rewards depend on this policy. The policy may be deterministic or random, i.e., the policy may dictate precisely what action to take at a point in the state space (a deterministic policy), or it may dictate only the distribution over actions to take at a point in the state space (a randomised policy).

The agent's value function associated with such a control problem is

$$V(x) = \max_{\pi \in \mathcal{A}} \mathbb{E}[R_t^\pi | x_t = x]$$

and does not depend on t due to the stationarity assumption. There is a slight abuse of notation here, as we use $\mathcal{A}$ to denote both the set of allowable actions and the set of policies – the context should it make clear which version is relevant. The dynamic programming principle implies that V satisfies the Bellman equation

$$V(x) = \max_{\pi \in \mathcal{A}} \mathbb{E}[r_t^\pi + \gamma V(x_{t+1}^\pi) | x_t = x].$$

In RL, it proves useful (as it leads to efficient learning algorithms) to define the action–value function which measures the "quality" of taking a specific action a, and hence is called the Q-function, as

$$Q(x, a) = \mathbb{E}[r_t^a + \gamma V(x_{t+1}^a) | x_t = x], \qquad x \in \mathcal{X}, a \in \mathcal{A}.$$

Here, a is an arbitrary action taken at time t, x_{t+1}^a denotes the one-step evolution of the state under action a, and r_t^a the corresponding reward. As $V(x) = \max_a Q(x, a)$, the action–value function satisfies a Bellman-like equation, namely

$$Q(x, a) = \mathbb{E}\left[r_t^a + \gamma \max_{a' \in \mathcal{A}} Q(x_{t+1}^a, a') \middle| x_t = x \right]. \tag{2.1}$$

This (fixed-point) equation is the starting point for a multitude of RL methods, including e.g. Monte Carlo methods, policy iteration, state–action–reward–state–action (SARSA), and Q-learning (see Sutton and Barto [109, Chaps. 4–6] for an overview of these methods).

2.1 Tabular Q-learning

One of the most straightforward approaches to solving the Bellman equation (2.1) is tabular Q-learning – which refers to the case when the action and state space are either discrete or, if continuous, approximated to be discrete. In this case, the Q-function is simply a table of values (see Table 1) where, e.g., each row represents the quality of all possible actions for a particular state.

Q-learning aims to estimate $Q(x, a)$ directly from observations of 4-tuples $(x_t, a_t, r_t^{a_t}, x_{t+1}^{a_t})$. Under certain conditions, stochastic approximation methods show that the iterative update scheme (see Watkins and Dayan [118], Tsitsiklis [112])

$$Q(x_t, a_t) \leftarrow Q(x_t, a_t) + \alpha_t \Big(\big(r_t^{a_t} + \gamma \max_{a'} Q(x_{t+1}^{a_t}, a') \big) - Q(x_t, a_t) \Big), \tag{2.2}$$

Table 1 Tabular Q-function is a table of quality of actions for particular states

$Q(x_1, a_1)$	$Q(x_1, a_2)$	$Q(x_1, a_3)$
$Q(x_2, a_1)$	$Q(x_2, a_2)$	$Q(x_2, a_3)$
$Q(x_3, a_1)$	$Q(x_3, a_2)$	$Q(x_3, a_3)$
$Q(x_4, a_1)$	$Q(x_4, a_2)$	$Q(x_4, a_3)$

```
   Initialise: initialise $Q(\cdot,\cdot)$ and replay buffer $\mathcal{B}$
1  repeat
2      observe state $x_\tau$;
3      with probability $\varepsilon$, select $a_\tau \in \mathcal{A}$ at random, otherwise set $a_\tau = \operatorname{argmax}_{a\in\mathcal{A}} Q(x_\tau, a)$;
4      take action $a_\tau$;
5      receive reward $r_\tau^{a_\tau}$ and new state $x_{\tau+1}$;
6      add $(x_\tau, a_\tau, r_\tau^{a_\tau}, x_{\tau+1})$ to replay buffer $\mathcal{B}$;
7      update Q-function $Q(x_\tau, a_\tau) \leftarrow Q(x_\tau, a_\tau) + \alpha_\tau \Big(r_\tau^{a_\tau} + \gamma \max_{a'\in\mathcal{A}} Q(x_{\tau+1}^{a_\tau}, a') - Q(x_\tau, a_\tau)\Big)$;
8      for M iterations do
9          sample $(x_t, a_t, r_t^{a_t}, x_{t+1})$ from $\mathcal{B}$;
10         update Q-function
             $Q(x_t, a_t) \leftarrow Q(x_t, a_t) + \alpha_t \Big(r_t^{a_t} + \gamma \max_{a'\in\mathcal{A}} Q(x_{t+1}^{a_t}, a') - Q(x_t, a_t)\Big)$;
11     end
12 until converged;
13 return $Q(\cdot,\cdot)$
```

Algorithm 1: The Dyna-Q learning algorithm

with $\alpha_t > 0$, $\sum_t \alpha_t = \infty$, $\sum_t \alpha_t^2 < \infty$, converges to the correct fixed point. The arrow in (2.2) represents replacing the element on the left with the value on the right. This iterative scheme should be understood as taking (sequential) samples of states x_t and, taking what is known as ε-greedy actions a_t, observing the resulting reward $r_t^{a_t}$ and the new state $x_{t+1}^{a_t}$ that the system evolves into, and then updating $Q(x_t, a_t)$ according to (2.2). An ε-greedy action consists of sampling a Bernoulli random variable H (with success probabilities that tend to zero as the number of iterations tends to infinity) and if $H = 1$, selecting a random action from the allowable set of actions, if $H = 0$, selecting an optimal action from the current estimate of the Q-function, i.e., $a_t = \operatorname{argmax}_{a'\in\mathcal{A}} Q(x_t, a')$. An alternative to the basic ε-greedy strategy is to sample actions using Boltzmann weights $e^{Q(x_t,a)} / \sum_{a'\in\mathcal{A}} e^{Q(x_t,a')}$ over actions. Such a randomisation of actions is key to RL methods as it allows the agent to explore the state space while simultaneously exploiting what has been learnt to be optimal thus far – the so-called exploration/exploitation tradeoff which appears again and again.

This basic form of Q-learning updates the Q-function at each state–action pair only whenever that state–action pair is visited. As a result, it tends not to work very well, and there are many improvements in the extant literature. One simple but effective improvement is to use the Dyna-Q learning approach which employs a replay buffer. A replay buffer consists of histories of the set of states, action and rewards $(x_t, a_t, r_t^{a_t}, x_{t+1})$, called a 4-tuple. In the simplest form of Dyna-Q learning, the update rule (2.2) is applied to samples from the replay buffer in addition to realisations. This resampling results in improved estimates of the Q-function at previously visited state–action pairs. Algorithm 1 summarises the steps of the approach.

When the state and action space cannot be made approximately discrete, or dimensions are large, tabular Q-learning and its Dyna-Q learning cousin fail. In these settings, function approximation approaches, including deep Q-networks (DQN) (see Mnih et al. [88]), double DQN (DDQN) (see van Hasselt et al. [114]), and dueling DQN (see Wang et al. [117]) are utilised and we discuss these approaches next.

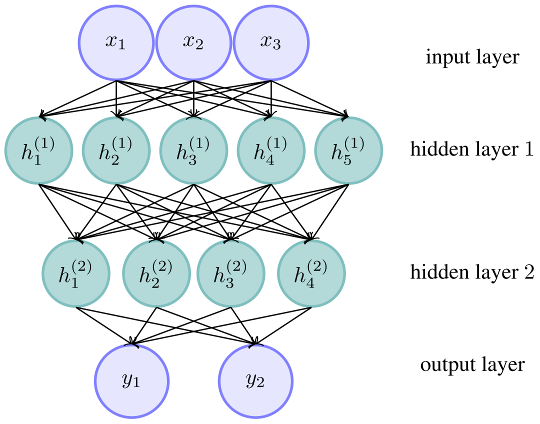


Fig. 2 A schematic of a feed-forward ANN

2.2 Deep Q-networks

Before delving into the details of deep Q-networks, it is instructive to briefly introduce artificial neural networks (ANNs, also known simply as NNs). Cybenko [35], Hornik et al. [62], Hornik [61] have shown that ANNs are universal function approximators, i.e., ANNs can approximate arbitrarily well any continuous function with compact support. The proofs, however, are not constructive and do not provide the ANN that achieves a particular accuracy for a given target function. Nonetheless, ANNs have proved exceedingly useful. An ANN is simply a sequence of nonlinear maps from inputs to outputs, constructed from affine transformations and so-called activation functions; see Fig. 2.

The intermediate output values at each step in the sequential map are called hidden layers. Letting $h^{(k)}$ denote the values at hidden layer k ($k = 0$ denotes the input layer and $k = K$ the output layer), the nonlinear map may be written sequentially as $h^{(k)} = (a^{(k)} \circ p^{(k)})(h^{(k-1)})$, $k = 1, \dots, K$, where $p^{(k)}$ denotes an affine transformation and $a^{(k)}$ denotes an activation function[1] for layer k. Deep learning (DL) approaches use ANNs for various modelling ingredients. The term "deep" in DL refers to having many hidden layers. The term "learning" in DL refers to estimating the network's parameters, consisting of the weights and biases that form the affine transformation between layers (when learning, the activation functions and architecture of the ANN are held fixed).

At this point, it is worth commenting that while proponents pitch DL approaches as being "model-free", this is not a wholly fair or transparent statement. The "model" in DL approaches is the ANN architecture used to connect inputs to outputs. There are numerous architectures (for example a feed-forward, a recurrent neural or a convolution neural net, etc.), and even when the general framework is specified, there are a host of "hyper-parameters" that require tuning – how many nodes in a layer, how many layers, what is the learning rate, what optimiser is being used, and so on.

[1]Typical activation functions are sigmoids $a(x) = (1 + e^{-x})^{-1}$, rectified linear units (RELU) $a(x) = x^+$, and softplus $a(x) = \log(1 + e^x)$, but there are numerous others.

Deep Q-networks (DQN) (see Mnih et al. [88]) are a form of function approximation for the Q-function where an ANN parametrises it, and we write $Q(x,a;\theta)$, with θ denoting the network parameters. Rather than using the stochastic approximation update rule (2.2), the update rule minimises a loss function that aims to enforce the Bellman equation (2.1). Specifically, the loss function over a mini-batch $\mathcal{B}$ of data is

$$L(\theta)=\sum_{i\in\mathcal{B}}\Big(\Big(r_i^{a_i}+\gamma\max_{a'\in\mathcal{A}}Q(x_{i+1}^{a_i},a';\theta)\Big)-Q(x_i,a_i;\theta)\Big)^2. \tag{2.3}$$

The network is updated by taking a so-called (stochastic) gradient descent[2] (GD) step $\theta\leftarrow\theta-\eta\nabla_\theta L(\theta)$, sampling a new mini-batch, and repeating until the parameters converge. The term "stochastic" GD refers to the fact that only a mini-batch of data, rather than all data, is being used to compute the gradients and hence update the parameters. The hyper-parameter η is called the learning rate and should be chosen judiciously. In practice, different learning rates in powers of ten are tested and the one which results in the best convergence is selected. Often, the learning rate is tuned downwards as the iteration progresses. The gradient $\nabla_\theta L(\theta)$ is obtained via a computationally efficient method known as backpropagation (see e.g. Chauvin and Rumelhart [32, Chap. 1]). The convergence of a slightly modified DQN approach (known as fitted Q-iteration (FQI), see Riedmiller [100]) is established in Fan et al. [40].

The loss function (2.3) often leads to over-estimates of the Q-function, resulting in sub-optimal actions. To circumvent this issue, double DQN (DDQN), see van Hasselt et al. [114], utilises the idea of a target network ϑ and main network θ, and instead minimises (over θ) the loss

$$L(\theta;\vartheta)=\sum_{i\in\mathcal{B}}\Big(\big(r_i^{a_i}+\gamma\,Q(x_{t+1}^{a_i},a_i^*;\vartheta)\big)-Q(x_i,a_i;\theta)\Big)^2, \tag{2.4}$$

where

$$a_i^*=\operatorname*{argmax}_{a'\in\mathcal{A}}Q(x_{t+1}^{a_i},a';\theta).$$

In this manner, the optimal action is chosen using the main network θ, while the "quality" of that action is evaluated using the target network ϑ. After several mini-batch GD steps, the target network is replaced by the main network and the process continues until it converges. This approach has been shown to increase stability in the optimal strategies.

As a demonstration of the applicability of these methods to financial problems, we show results from Cartea et al. [25], where the authors implement DDQN for optimal trading in foreign exchange (FX) triplets. An FX triplet consists of exchange rates between three currencies, e.g. EURUSD, GBPUSD, EURGBP. The FX triplets

[2]There are many other optimisation update rules such as momentum (see Polyak [97], Sutskever et al. [108]) (which exponentially smooths the gradients) and Adam (see Kingma and Ba [70]) (a combination of momentum and root-mean-squared propagation).

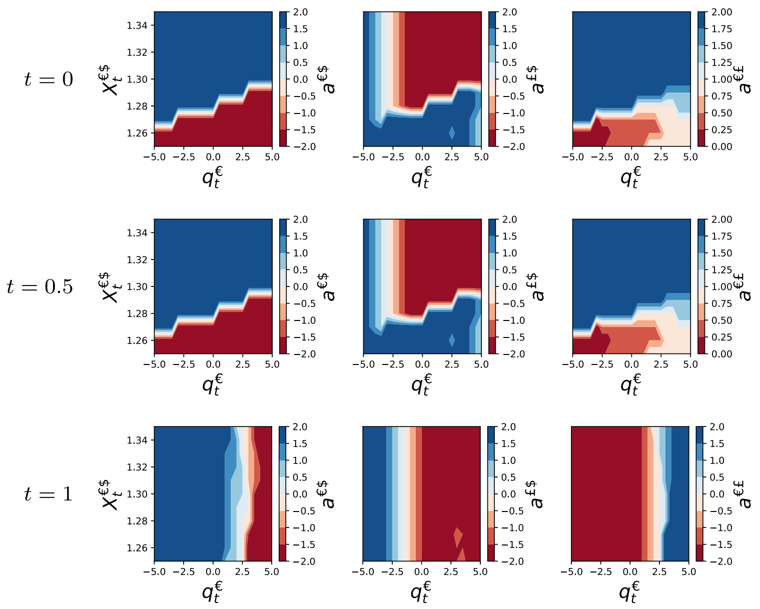


Fig. 3 Optimal trading strategy in FX triplets using DDQN over one day. Each column corresponds to the optimal action in one of the triplet pairs, as a function of inventory in € and €$ FX rate while the €£ rate and position in £ are held fixed

are cointegrated and trading frictions prevent the trader from instantly changing positions. This corresponds to a system with five state variables, and even if the state dynamics are known, a numerical PDE approach to the stochastic control problem would not be feasible. Figure 3 (modified from [25]) shows that the DDQN algorithm, while not being aware of the FX dynamics, learns to take advantage of cointegration while properly managing inventories in all three currencies. Other examples of DDQN in financial modelling include Ning et al. [93], Dabérius et al. [36], Kumar [74]. Applications of path signatures and ML to an optimal execution problem may be found in Cartea et al. [26].

Next, the dueling DQN approach (see Wang et al. [117]) breaks the action–value function into two terms – the value function and the advantage function – and models each with separate ANNs. Explicitly,

$$Q(x, a; \alpha, \beta) = V(x; \alpha) + A(x, a; \beta),$$

with network parameters α and β. Often the average of A over a is subtracted to aid with identifiability. Using the above form of the Q-function, the loss function becomes

$$L(\alpha, \beta) = \sum_{i \in \mathcal{B}} \left(\left(r_i^{a_i} + \gamma \max_{a' \in \mathcal{A}} A(x_{i+1}, a'; \beta) - A(x_i, a_i; \beta) \right) + \left(\gamma\, V(x_{i+1}; \alpha) - V(x_i; \alpha) \right) \right)^2 .$$

Learning proceeds by taking mini-batch GD steps by first freezing the advantage network β and "learning" the value network α, and then freezing the value network α and "learning" the value network β, over several iterations. Then, one samples a new mini-batch and repeats the process. The term dueling networks stems from the presence of the two networks and that learning proceeds by minimising the loss while one network is held fixed. This separation of the Q-function in terms of value and advantage functions will reappear when we study multi-agent games in Sect. 6.

The dueling framework may be viewed as a sort of generalised adversarial network (GAN); see Goodfellow et al. [50]. GANs are essentially min–max games where an actor generates data from a model, and a critic assesses how close that generated data is to test data. More specifically, GANs may be viewed as a game between two players, where one player (the actor) controls a generator G_θ and aims to minimise a loss L_G and a second player (the critic) controls a discriminator D_ϑ and aims to minimise a loss L_D. Each loss function depends on the other's control. More specifically, the goals may be written as finding

$$\theta^* \in \operatorname*{argmin}_{\theta\in\Theta} L_G(\theta, \vartheta^*) \quad \text{and} \quad \vartheta^* \in \operatorname*{argmin}_{\vartheta\in\varTheta} L_D(\theta^*, \vartheta).$$

The dueling DQN approach may thus be seen as a GAN where the loss functions are identical and the networks for advantage and value function are the actor and critic.

2.3 Policy gradient methods

The DQN-based approaches require seeking for $\operatorname{argmax}_{a\in\mathcal{A}} Q(x, a; \theta)$ and therefore become infeasible when the action space is itself continuous. To circumvent this issue, policy gradient (PG) methods were developed in Silver et al. [104], and extended to deterministic policy gradient (DPG) methods in Lillicrap et al. [84]. DPG is based on the following set of observations. First, with $F(x') := \mathbb{P}[x_1 \le x']$, define the (un-normalised) discounted state distribution from following the policy π as

$$\rho^\pi(x) = \int_{\mathcal{X}} \sum_{t=1}^{\infty} \gamma^{t-1} \mathbb{P}[x_t = x | x_1 = x', \pi]\, dF(x').$$

Then the value function $V(\pi)$ associated with a policy π, after taking the expectation over the initial state, may be written as $V(\pi) = \mathbb{E}^\pi[r(X, a^\pi(X))]$ (note this is the expectation of the one-step reward r), where $\mathbb{E}^\pi[\,\cdot\,]$ means expectation over the (un-normalised) discounted state distribution ρ^π, where $X \sim \rho^\pi$. If we parametrise the (deterministic) policy by θ, we may write $V(\pi_\theta) = \int_{\mathcal{X}} \rho^{\pi_\theta}(x) r(x, a(x; \theta))\, dx$. DPG uses a GD step on policies and updates the policy in the direction of $\nabla_\theta V(\pi_\theta)$. Sutton et al. [110] show that

$$\nabla_\theta V(\pi_\theta) = \int_{\mathcal{X}} \rho^{\pi_\theta}(x) \nabla_\theta Q\big(x, a(x; \theta); \theta\big)\, dx = \mathbb{E}^{\pi_\theta}\big[\nabla_\theta Q\big(x, a(x; \theta); \theta\big)\big]. \tag{2.5}$$

The surprising aspect of this result is that while the (un-normalised) discounted state distribution ρ^{π_θ} depends on the policy parameters, its gradients do not show up in the value function's gradient. When the policy is parametrised by a deep ANN, the procedure is known as deep DPG (DDPG).

This observation is often used in tandem with actor–critic methods with four networks: θ, θ' determine the policies (the actor and actor's target), and ϑ, ϑ' determine the Q-functions (the critic and critic's target). The RL algorithm proceeds as follows. Observe a mini-batch of states $(x_t)_{t\in\mathcal{B}}$, take actions $(a_t = a(x_t;\theta))_{t\in\mathcal{B}}$ (sometimes noise is added to allow additional exploration), observe the new states $(x_{t+1})_{t\in\mathcal{B}}$ and rewards $(r_t)_{t\in\mathcal{B}}$. Use these 4-tuples to update the critic's network ϑ by taking a GD step with loss function

$$L(\vartheta;\theta',\vartheta') = \sum_{t\in\mathcal{B}} \Big(\big(r_t + \gamma\, Q\big(x_{t+1}, a(x_{t+1};\theta');\vartheta'\big)\big) - Q(x_t, a_t;\vartheta)\Big)^2,$$

and update the actor's policy network θ using DPG via

$$\theta \leftarrow \theta - \frac{\eta}{N}\sum_{t\in\mathcal{B}} \nabla_\theta\, Q\big(x_t, a(x_t;\theta);\vartheta\big),$$

where the summation equals the mini-batch sample average of $\mathbb{E}\,[\nabla_\theta V(\pi_\theta)]$ in (2.5). The target networks are updated according to the rules $\theta' \leftarrow \alpha\,\theta + (1-\alpha)\,\theta'$ and $\vartheta' \leftarrow \alpha\,\vartheta + (1-\alpha)\,\vartheta'$, which allows the target networks to slowly progress towards the main networks.

One immediate application area for DDPG are portfolio optimisation problems in settings where analytical methods fail (even if the model is fully specified). For example, one can consider including investment goals, see Nevins [90] (which are essentially probability constraints), or encoding stochastic benchmarks, as in Al-Aradi and Jaimungal [2, 3] (that make assumptions so that solutions are in closed form) or Ni et al. [91] who use ANNs to parametrise strategies but perform optimisation using straight forward backpropagation and mini-batch GD updates.

As an explicit example, let us consider Pesenti and Jaimungal [95] who investigate how to minimise a distortion risk measure (e.g. conditional value-at-risk) on terminal wealth subject to two constraints: (i) strategies are self-financing and satisfy a budget constraint, and (ii) strategies result in terminal wealth distributions that lie within a Wasserstein ball around a given benchmark's terminal wealth. That is, they seek

$$\begin{aligned} &\min_{\pi\in\mathcal{A}} -\int_0^1 F^{-1}_{X^\pi_T}(u)\,\gamma(u)\,du \\ \text{subject to}\quad &\inf_{\chi\in\Pi(F_{X^\pi_T},F_{X^\delta_T})} \left(\int_{\mathbb{R}^2} |y_1-y_2|^2\,\chi(dy_1,dy_2)\right)^{\frac{1}{2}} \le \varepsilon, \end{aligned}$$

where X^δ is a benchmark portfolio, $\gamma : [0,1] \to \mathbb{R}_+$ a distortion weight function (with $\int_0^1 \gamma(u)\,du = 1$), and $\Pi(F_1, F_2)$ denotes all couplings between distribution functions F_1 and F_2. This problem is far more challenging than outright portfolio allocation. Nonetheless, DDPG methods can be applied, and Fig. 4 shows the results of an implementation when asset prices follow a stochastic volatility model. Even

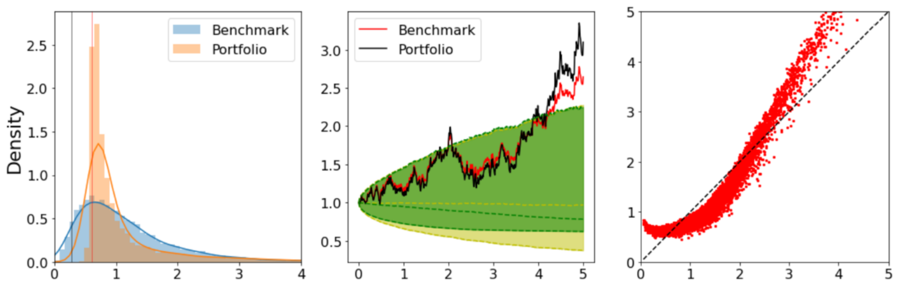


Fig. 4 Example portfolio optimisation where the goal is to minimise a particular distortion risk measure while remaining within a Wasserstein ball around a given benchmark

in this rather complex environment, the approach is able to improve the risk measure from -10.1 to -14.23 while respecting the constraint, and produces financially sound results. For further examples and generalisations see [69] who develop a deep learning approach for solving a wide class of robust risk-averse reinforcement learning problems.

3 Exploratory controls

In Wang et al. [115] and Wang and Zhou [116], the authors introduce a line of work that aims to show why the exploration–exploitation tradeoff is valuable for RL problems (in continuous time and space). They suppose that agents control the distribution over actions (akin to the randomised strategies discussed in the previous sections) in a manner that is modulated by time and state, i.e., the agent controls a flow of measures π_t that is Markov in the state. Further, if the "deterministic" control problem has a state evolution (under an action process u)

$$dx_t = b(x_t, u_t)\,dt + \sigma(x_t, u_t)\,dW_t,$$

then the relaxed control problem (as the randomised policy version of the problem is called) satisfies the SDE

$$dx_t = \int_{\mathcal{A}} b(x_t, u)\pi_t(u)\,du\,dt + \sqrt{\int_{\mathcal{A}} \sigma^2(x_t, u)\,\pi_t(u)\,du}\,dW_t.$$

This has the interpretation of the instantaneous drift and variance being randomised over the action distribution. The agents' performance criterion is also modified to include an entropy term, so that

$$V(x) = \inf_{\pi\in\mathcal{D}} \mathbb{E}\left[\int_0^\infty e^{-\rho t} \int_{\mathcal{A}} \big(f(x_t, u) + \lambda \log \pi_t(u)\big)\pi_t(u)\,du\,dt \,\middle|\, x_t = x\right].$$

The function f plays the usual role of a running cost, while the second term is an entropy term that encourages exploration. Interestingly, in the linear–quadratic

Gaussian (LQG) case, Wang et al. [115] demonstrate that the optimal measure flow is Gaussian where (i) the mean equals the optimal solution in the deterministic policy case (states are mapped uniquely to actions as in classical stochastic control), and (ii) the standard deviation is proportional to λ. Hence, the strength of the entropy term controls variability around the control that the agent would deem to be optimal if they did not explore.

These results are generalised to mean-field game (MFG) settings in Guo et al. [55] and Firoozi and Jaimungal [46]. The latter also studies the heterogeneous agents case, provides a proof of the ε-Nash equilibria property for the exploratory problem, and establishes that the existence of the solutions to the "classical" and exploratory LQG MFG systems are equivalent.

None of these works, however, provide a basis for a model-free approach to exploration/exploitation. One idea perhaps worth exploring is to consider a batch RL paradigm, whereby an LQG model is learnt from observing the environment, the agent employs the exploratory optimal control, and then updates the estimated LQG model with new data, applies the new exploratory optimal controls, updates the estimated LQG model with new data, and so on. One issue with this approach is that the real systems are likely not LQG, and while the procedure may converge, it need not provide near-optimal solutions. Alternatively, the ideas in the next section – where a fully nonlinear system is viewed as an LQG system along perturbed sub-optimal paths that are then updated – together with the exploratory control approaches here may prove a viable alternative.

4 Iterative LQG control

The iterative linear quadratic regulator (iLQR) introduced in Li and Todorov [81] looks at deterministic control problems, where the state equation is known but nonlinear, and provides update rules to improve upon a nominal control by performing an expansion on the state along the previous (sub-optimal) trajectory.

These ideas were extended in Todorov and Li [111] to stochastic control problems as follows. Suppose the controlled state satisfies the SDE

$$dx_t = f(x_t, u_t)\,dt + \sigma\,dW_t$$

and the agent aims to minimise the performance criterion

$$J(u) = \mathbb{E}\left[\int_0^T g(x_t, u_t)\,dt\right], \tag{4.1}$$

where f and g are nonlinear. The focus of this example is on a controlled drift with running costs alone. The extension to controlled volatility and a terminal cost could follow along similar lines of thought.

Next, [111] supposes that we have a current estimate of the "optimal" control $\bar{u}$ and the corresponding controlled state $\bar{x}$ (obtained by ignoring the diffusion term). We aim to improve upon this estimate by expanding around the estimate and solving a new linear–quadratic problem.

To this end, write $x_t = \bar{x}_t + \delta x_t$ and $u_t = \bar{u}_t + \delta u_t$ and consider δu as the new control (where $\bar{u}$ is held fixed) that controls δx. Then expand to first order,

$$f(\bar{x}_t + \delta x_t, \bar{u}_t + \delta u_t) \approx f(\bar{x}_t, \bar{u}_t) + \big(\nabla f(\bar{x}_t, \bar{u}_t)\big)^\top \delta y_t,$$

and similarly expand to second order,

$$g(x_t, u_t) \approx g(\bar{x}_t, \bar{u}_t) + \big(\nabla g(\bar{x}_t, \bar{u}_t)\big)^\top \delta y_t + \frac{1}{2}\delta y_t^\top \mathsf{H} g(\bar{x}_t, \bar{u}_t)\, \delta y_t,$$

where $\delta y_t = (\delta x_t, \delta u_t)^\top$ and $\mathsf{H}g$ denotes the Hessian of g. Next, the SDE for δx may be written as

$$d(\delta x_t) = (\bar{f}_t + \bar{f}_{x,t}\, \delta x_t + \bar{f}_{u,t}\, \delta u_t)\, dt + \sigma\, dW_t - d\bar{x}_t,$$

where $\bar{f}_t = f(\bar{x}_t, \bar{u}_t)$, $\bar{f}_{x,t} = \partial_x f(\bar{x}_t, \bar{u}_t)$ and $\bar{f}_{u,t} = \partial_u f(\bar{x}_t, \bar{u}_t)$, which correspond to the drift along the previous approximately optimal path, and its sensitivity to state and control. As the state dynamics is affine in the modified state and control (δx and δu) and the running cost is at most quadratic in the modified state and control, the optimal modification of the control will be affine,[3] so that we may write

$$\delta u_t^* = a_t + b_t\, \delta x_t^*$$

for some (known) stochastic processes (a_t) and (b_t) that depend on the previous estimate of the "optimal" control and corresponding path ($\bar{x}$ and $\bar{u}$) – obtained e.g. by solving the dynamic programming equation, a linear forward–backward SDE (FBSDE) stemming from the stochastic Pontryagin maximum principle, or a variational analysis approach to optimising (4.1) with the perturbed dynamics and running cost. Thus, the process

$$u_t^* = \bar{u}_t + \delta u_t^*$$

provides an improved control and we may iterate the procedure until it converges.

This iterative procedure appears to work well in several settings, and when the nonlinear dynamics is not known, approaches for estimating approximate linear dynamics along observed paths have been suggested in e.g. Gu et al. [53]. It would be interesting to further develop these approaches for applications in finance, e.g. propagator models of nonlinear impact (see Gatheral [47]), trading signals in Cartea and Jaimungal [24], Lehalle and Neuman [80], and latent factors in Casgrain and Jaimungal [28]. The idea of coupling variants of iLQG with exploratory controls as outlined in Sect. 3 and its generalisation to MFGs (see Guo et al. [55], Firoozi and Jaimungal [46]) is intriguing. Making all of these procedures precise, and finding conditions under which the algorithm converges to the true solution, is a challenging but fruitful line of research that would go a long way.

[3] This is despite the fact that the perturbed model is not linear–quadratic due to the presence of the stochastic path from the previous iteration of the control appearing in both the state dynamics and the cost functional.

5 Deep BSDE for stochastic control

Stochastic control problems are intimately connected with FBSDEs, and this section briefly discusses deep learning approaches for solving such equations. In settings where the dynamics of the stochastic control system is known, the deep BSDE approach (see E et al. [39], Han and Long [57]) has been shown to be a successful alternative to least-squares-regression approaches. The idea is as follows. Suppose that the state process satisfies the SDE

$$dx_t = b\big(t, x_t, \alpha(t, x_t)\big)\, dt + \sigma(t, x_t)\, dW_t,$$

where α represents the agent's control, and the agent has the performance criterion

$$J^\alpha(t, x) = \mathbb{E}\bigg[\int_t^T f\big(s, x_s, \alpha(s, x_s)\big)\, ds + g(x_T)\bigg]$$

and value function $J(t, x) = \min_\alpha J^\alpha(t, x)$. Then, under certain conditions (see e.g. Pham [96, Chap. 6]), the optimisation problem may be rewritten as the solution to the FBSDE

$$x_t = x_0 + \int_0^t b\big(s, x_s, \alpha^*(s, x_s)\big)\, ds + \int_0^t \sigma(s, x_s)\, dW_s, \tag{5.1}$$

$$y_t = g(x_T) + \int_t^T H(s, x_s, y_s)\, ds - \int_t^T z_s\, dW_s, \tag{5.2}$$

where $H(t, x, y) = \min_a (b(t, x, a)\, y + f(t, x, a))$ is the system's Hamiltonian, and we set $\alpha^*(t, x_t) = \operatorname{argmin}_a (b(t, x_t, a)\, y_t + f(t, x_t, a))$, $y_t = J(t, x_t)$, as well as $z_t = \sigma(t, x_t)\, \partial_x J(t, x_t)$. The deep BSDE approach to the optimisation problem proceeds by first performing an Euler discretisation of (5.1) and (5.2) leading to

$$\begin{aligned} x_{t_{k+1}} &= x_{t_k} + b\big(t_k, x_{t_k}, \alpha^*(t_k, x_{t_k})\big)\Delta t_k + \sigma(t_k, x_{t_k})\, \Delta W_{t_k}, \\ y_{t_{k+1}} &= y_{t_k} - H(t_k, x_{t_k}, y_{t_k})\Delta t_k + z_{t_k}\, \Delta W_{t_k}, \end{aligned}$$

for $0 = t_0 < t_1 < \cdots < t_N = T$. The method then assumes that $y_0 = Y(x_0; \theta)$ and $z_{t_k} = Z(x_{t_k}; \vartheta_k)$, $k = 0, \ldots, N-1$, where θ and ϑ_k parametrise ANN approximations for the mapping from state to adjoint (at time 0) and co-adjoint (at all discrete time points). For fixed ANNs, sample paths of x, y, z may be generated, and the goal is to minimise the loss function

$$L(\theta; \vartheta) = \frac{1}{M} \sum_{m=1}^{M} \big(y_T^{(m)} - g(x_T^{(m)})\big)^2$$

over θ and ϑ, which aims to enforce the terminal condition constraint.

Naturally, one may also take the PDE approach for solving control problems. An early work in this direction is Sirignano and Spiliopoulos [106], where the solution to the PDE is approximated by an ANN. For an overview and a number of financial applications, we refer to Al-Aradi et al. [1]. Germain et al. [48] provide an overview of a number of other approaches. See Huré et al. [68] for extensions to variational inequalities as well as reflected BSDEs and Saporito and Zhang [102] for extensions to path-dependent PDEs.

An interesting challenge is to extend these ideas to the case when the state dynamics themselves are not known. One simple idea is to introduce a prior on a set $\mathcal{P}$ of models, solve for the optimal controls for each model, using ε-greedy actions where the randomisation is over models using the prior distribution, make observations that update your prior, and repeat. This requires having a good set of models that you are confident reflects the environment; how to go beyond that and directly learn from the environment is a real challenge. Perhaps tying this into exploratory control in Sect. 3 and iLQG methods in Sect. 4 would prove useful. Also, making use of DPG methods (see Sect. 2.3) may prove helpful in accelerating parameter learning. A final direction could be to assume that the coefficients of the SDE are themselves parametrised by ANNs – so-called neural SDEs as in Li et al. [82], Gierjatowicz et al. [49] (see also Cuchiero et al. [34] for neural ODEs). Then solve for the optimal controls using the deep BSDE approach, take these optimal actions to the environment and observe over several epochs, after which you update the neural SDE and repeat.

6 Multi-agent games

Multi-agent games arise in many contexts in financial mathematics, including algorithmic trading (Firoozi and Caines [45], Casgrain and Jaimungal [27], Cardaliaguet and Lehalle [15], Huang et al. [67], Lehalle and Mouzouni [79], Casgrain and Jaimungal [29]), energy markets (Shrivats et al. [103], Féron et al. [43], Bouveret et al. [9]), and systemic risk (Carmona et al. [19], Bo and Capponi [7], Borovykh et al. [8]). Much research effort has been placed on developing the theory and application of mean-field games (MFGs) as a means to approximate Nash equilibria for multi-agent games by their infinite population limits. The MFG methodology was originally developed in Huang et al. [65, 66] and Lasry and Lions [76, 77, 78], and many extensions and generalisations exist, including the probabilistic approach to MFGs (Carmona and Delarue [17, 18]), MFGs with common noise (Cardaliaguet et al. [14]), major–minor MFGs (Huang [64]), and the master equation approach (Cardaliaguet et al. [14]).

As only very few MFGs are solvable in closed form, there has been a flurry of activity from the financial mathematics community in developing solutions to MFGs and mean-field type control (MFC) problems using variations of Q-learning adapted to the MFG and MFC settings, including Yang et al. [120], Carmona et al. [20], Carmona and Laurière [21], Carmona et al. [22, 23], Guo et al. [54], Gu et al. [51, 52]. Due to limited space, the details of these approaches are omitted, and this section instead focuses on approaches for solving finite-player, as opposed to the infinite-population-limit, versions of multi-agent games using RL.

A Q-learning-based approach for obtaining Nash equilibria in general-sum stochastic games first appears in Hu and Wellman [63]. They prove convergence of their algorithm for games with finite state and action spaces; however, their approach is computationally infeasible for all but the simplest examples. The main computational bottleneck is the need to repeatedly compute a local Nash equilibrium over states (a stage game), which is an NP-hard operation in general. Moreover, the method proposed in [63] does not extend to games where agents choose continuous-valued controls or to games with either high-dimensional states or with a large (but finite) number of players. Alternatively, Casgrain et al. [31] combine the iLQG framework of Todorov and Li [111], its actor–critic variation in Gu et al. [53] and the Nash Q-learning algorithm of Hu and Wellman [63] to produce an algorithm that can learn Nash equilibria in these more complex and practically relevant settings. First, in the finite-player setting, each agent has their own Q-function which is a function of the states of agents and their own control. The analogue of the Bellman equation is

$$Q_i(x; u) = \mathbb{E}[r_{i,t} + \gamma \boldsymbol{\mathcal{N}}_{u'} Q_i(x_{t+1}; u')],$$

where $\boldsymbol{\mathcal{N}}_{u'}$ represents the static stage-game operator and i enumerates agents. We can try a deep learning approach to solve for the fixed point and minimise a loss akin to (2.4); however, this is computationally infeasible due to the necessity of solving the stage game at each iteration and at all points in the action–state space. Thus, Casgrain et al. [31] propose to write the Q-function in terms of value functions $V_i(x; \vartheta)$ and advantage functions $A_i(x, u; \theta)$, where the advantage functions are linear–quadratic in controls, but generally nonlinear in states, e.g.

$$\begin{aligned} A_i(x, u; \theta) = &- \begin{pmatrix} u_i - \mu_i^\theta(x) \\ u_{-i} - \mu_{-i}^\theta(x) \end{pmatrix}^\top P_i^\theta(x) \begin{pmatrix} u_i - \mu_i^\theta(x) \\ u_{-i} - \mu_{-i}^\theta(x) \end{pmatrix} \\ &+ \left(u_{-i} - \mu_{-i}^\theta(x)\right)^\top \Psi_i^\theta(x). \end{aligned}$$

For each x, $P_i^\theta(x)$ is a positive definite matrix, $\mu_i^\theta(x)$, $\mu_{-i}^\theta(x)$ and $\Psi_i^\theta(x)$ are vectors, and all are parametrised by an ANN with parameters θ. This formulation allows the stage game to be solved in closed form (given the network parameters), and we have by construction that the strategy attaining $\boldsymbol{\mathcal{N}}_u Q(x, u; \theta, \vartheta)$ is $\mu(x)$ and at the equilibria $\boldsymbol{\mathcal{N}}_u Q(x, u; \theta, \vartheta) = V(x; \vartheta)$. Hence, the loss function (over a mini-batch $\mathcal{B}$) becomes

$$L(\vartheta, \theta) = \sum_{t \in \mathcal{B}} \left(V(x_t; \vartheta) + A(x_t, u_t; \theta) - r_t - \gamma V(x_{t+1}; \vartheta)\right)^2,$$

which may be optimised in an actor–critic fashion by alternating between GD steps in θ and ϑ.

To illustrate a financial application of these methods, we take an example from algorithmic trading with multiple agents under nonlinear trading impact; see Casgrain et al. [31]. The price dynamics are

$$\Delta S_t = \left(\kappa\,(\theta - S_t) + b \,\mathrm{sgn}\Big(\frac{1}{N}\sum_{i=1}^N \Delta q_{t,i}\Big) \sqrt{\left|\frac{1}{N}\sum_{i=1}^N \Delta q_{t,i}\right|}\right) \Delta T + \sigma \sqrt{\Delta T}\, \xi_t.$$

Here, $\Delta q_{t,i}$ represents agent i's trades (changes in inventory) at t and $\xi_t \sim \mathcal{N}(0, 1)$ represents the one-step noisy innovations. Figure 5 (modified from [31]) shows results of applying the algorithm to a system with 5 agents. The results demonstrate

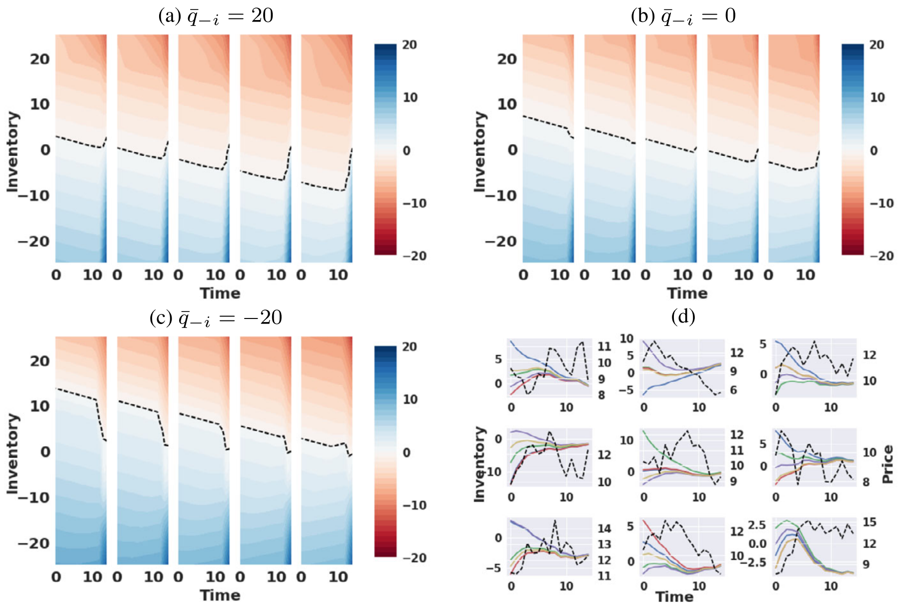


Fig. 5 Optimal trade execution heatmaps in (**a**)–(**c**) as a function of time, inventory, price and average inventory of other agents. Within each panel, subplots correspond to price levels \$6, \$8, . . . , \$14 from left to right. The dotted lines show the threshold where the agent switches from buying to selling. Panel (**d**) shows sample inventory paths and corresponding price paths. Solid lines represent agents' inventory paths, dotted lines represent asset price paths

that while the optimisation algorithm is unaware of the environment's dynamics, it finds an equilibrium that corresponds closely to the mean-field solution obtained in Casgrain and Jaimungal [27, 29] which generalise the results in Cartea and Jaimungal [24], Casgrain and Jaimungal [28]. The figure shows that agents, while ending the trading period with zero inventory, are attracted to a stochastic trend which depends on the asset price path, i.e., they are responding to price signals and account for the actions of aggregate trading.

Another approach, which requires solving at each stage two BSDEs, is deep fictitious play in Han and Hu [56] (which is proved to converge in Han and Long [57]). The central idea is as follows:

1. Make a (Markov) initial assumption on the optimal strategy for all players.
2. Each player optimises against this (fixed) estimated strategy. This step is solved using a deep FBSDE approach (as described in Sect. 5).
3. Players update their value functions according to the previous estimate of the optimal strategy.
4. Players update their policy by minimising the Hamiltonian using these new estimates.

Once again, it may be possible to integrate the ideas of "model-free" learning described in this section, or iterative methods (as in Sect. 4) together with deep fictitious play, to go beyond requiring the specification of model dynamics and to learn directly from data. Also, extending the ideas in Carmona et al. [20], Carmona and Laurière [21], Carmona et al. [22, 23], Gu et al. [51, 52], Guo et al. [54] to cases with major–minor agents and/or common noise, and graphon models of MFGs (see Caines and Huang [12], Carmona et al. [16]) would be very interesting. As well, recently, [13] look at principal-agent mean field games using deep learning techniques.

7 Reinforced deep Markov models

The approaches described thus far do not rely on specific models of data. There are many situations, however, where "model-driven" approaches have proved to be advantageous. The term model-driven here does not mean e.g. a particular SDE for the system dynamics, but rather a flexible DL model with additional structure that reflects aspects of the system being analysed without being too prescriptive. Kurutach et al. [75] argue that when deep neural networks are used to learn both the model and the policy, the learnt policy tends to exploit regions where there is actually little data, and thus induces instability in training. Instead, the authors propose to utilise an ensemble of models, which simultaneously accounts for uncertainty and regularises the problem. This section looks at one class of models known as deep Markov models (DMMs) that, as they contain features that resemble stochastic volatility as well as latent factors (both important aspects of financial data), are poised to be useful for financial modelling.

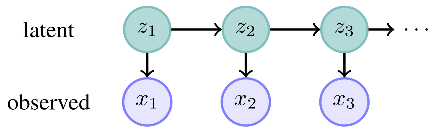
DMMs (see Krishnan et al. [72]) are generalisations of hidden Markov models (HMMs) to include nonlinear state-dependence between the latent (unobserved) states themselves and between latent states and observables. They may be viewed as in the graphical model in Fig. 6. For example, Gaussian DMMs (see Krishnan et al. [73]) make the assumption that

$$z_{t+1} \overset{\mathbb{P}}{\sim} \mathcal{N}\big(\mu_z^\theta(z_t), \Sigma_z^\theta(z_t)\big), \tag{7.1}$$

$$x_t \overset{\mathbb{P}}{\sim} \mathcal{N}\big(\mu_x^\theta(z_t), \Sigma_x^\theta(z_t)\big), \tag{7.2}$$

where x, z are visible and latent states, respectively, and the means and covariance matrices are parametrised by ANNs with parameters θ. If the means are affine and the covariances constant, the setup reduces to a Kalman filter model; hence the nonlinear mapping in a DMM distinguishes it from a Kalman filter model. The emission distribution (i.e., that of $x_t|z_t$) may be chosen to suit the needs of the modelling setting; e.g., if x is categorical, then a conditional multi-class logistic model parametrised by an ANN, $\mathbb{P}[x_t = i|z_t] = e^{w_i^\theta(z_t)} / \sum_{k=1}^K e^{w_k^\theta(z_t)}$, is useful, or if x is continuous and has support on $\mathbb{R}_+$, then a conditional Gamma distribution with shape $a^\theta(z_t)$ and scale $b^\theta(z_t)$ is useful, etc.

Fig. 6 Directed graph representation of a DMM



$$\begin{tikzcd} z_1 \arrow[r] \arrow[d] & z_2 \arrow[r] \arrow[d] & z_3 \arrow[r] \arrow[d] & \cdots \\ x_1 & x_2 & x_3 & \end{tikzcd}$$

As z is not visible, learning in such models requires the posterior distribution of z given observations x. This is generally intractable in such nonlinear models, and instead variational approximation (VA) techniques (see e.g. Ormerod and Wand [94] for an overview) are often employed. The essential idea behind VA techniques is to approximate an intractable posterior distribution with a parametrised (but rich enough) alternative distribution $\mathbb{Q}$ (which should be viewed as a distribution over $z_{1:T}$ conditional on observations $x^{\text{obs}}_{1:T}$) and maximise what is called the evidence lower bound (ELBO) (also known as the variational lower bound)

$$\mathcal{L}(x^{\text{obs}}_{1:T}) = \mathbb{E}^{\mathbb{Q}}\big[\mathbb{P}[x_{1:T} = x^{\text{obs}}_{1:T} | z_{1:T}]\big] - \mathbb{E}^{\mathbb{Q}}\bigg[\log \frac{d\mathbb{Q}}{d\mathbb{P}}\bigg].$$

Here and in the remainder of this section, we use the slice notation where $x_{1:T}$ denotes the vector $x_1, x_2, \dots, x_T$ and similarly for z, x^{obs}. A simple application of Jensen's inequality shows that the ELBO provides a lower bound for the log-likelihood, and when the relative entropy vanishes, the ELBO equals the log-likelihood. An example of a VA for the Gaussian DMM is to assume an encoder

$$z_{t+1} \overset{\mathbb{Q}}{\sim} \mathcal{N}\big(\mu^{\vartheta}_z(z_t, x_{1:T}), \Sigma^{\vartheta}_z(z_t, x_{1:T})\big), \tag{7.3}$$

where ϑ is a new ANN that parametrises the mean and covariance and takes the previous latent state and all observations as an input. Using all visible inputs is similar to the Kalman smoother. Instead of using the whole path of x, we can also use summaries (which is more memory-efficient) of the path of x up to time t, e.g. the output of a forward–backward long-short-term memory network (LSTM) or a gated recurrent unit (GRU). This shows the flexibility in DMM architectures. For the VA in (7.3), and in many other cases, the ELBO may be written out explicitly as

$$\begin{aligned}
&\mathcal{L}(x^{\text{obs}}_{1:T})\\
&= -\frac{1}{2}\sum_{t=0}^{T-1} \mathbb{E}^{\mathbb{Q}}\bigg[d_x \log(2\pi) + \log\det \Sigma^{\theta}_x(z_{t+1})\\
&\qquad + \big(x^{\text{obs}}_{t+1} - \mu^{\theta}_x(z_{t+1})\big)^{\top}\big(\Sigma^{\theta}_x(z_{t+1})\big)^{-1}\big(x^{\text{obs}}_{t+1} - \mu^{\theta}_x(z_{t+1})\big)\\
&\qquad + \log \frac{\det \Sigma^{\theta}_z(z_t)}{\det \Sigma^{\vartheta}_z(z_t, x^{\text{obs}}_{1:T})} + d_z - \operatorname{Tr}\Big(\big(\Sigma^{\theta}_z(z_t)\big)^{-1}\Sigma^{\vartheta}_z(z_t, x^{\text{obs}}_{1:T})\Big)\\
&\qquad - \big(\mu^{\theta}_z(z_t) - \mu^{\vartheta}_z(z_t, x^{\text{obs}}_{1:T})\big)^{\top}\big(\Sigma^{\theta}_z(z_t)\big)^{-1}\big(\mu^{\theta}_z(z_t) - \mu^{\vartheta}_z(z_t, x^{\text{obs}}_{1:T})\big)\bigg].
\end{aligned}$$

Learning proceeds in an actor–critic fashion by simulating mini-batches of paths from (7.3) (with ϑ frozen), taking GD steps in θ over several mini-batches, and then freezing ϑ and taking GD steps over ϑ, to maximise the ELBO and hence (hope) to maximise the log-likelihood.

The DMM may be used to develop models for sequential data and account for latent factors that modulate the dynamics. Such an approach is very pertinent to financial modelling, and it would be interesting to develop e.g. interest rate term structure models, models of volatility indices, volatility surfaces, FX rates, and so on.

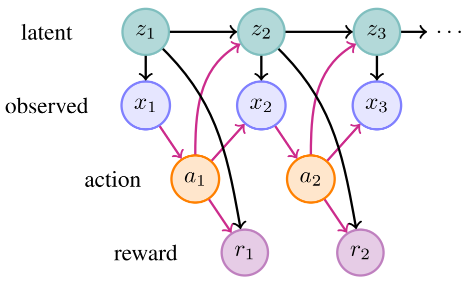


Fig. 7 Directed graph representation of a reinforced deep Markov model

The DMM setup was not initially envisioned for controlled systems. However, one modification (along the lines of Ferreira [44] who studies an optimal execution problem) that allows controls is to include actions and rewards as part of the graphical model as in Fig. 7. Such models are termed reinforced DMM (RDMM). RDMMs contain three network parameters θ, φ and ϑ. The first is the generative network θ in the DMM (7.1), (7.2) – which is now supplemented by a reward output and the effect of actions on the latent and visible states via

$$\begin{aligned} z_{t+1} &\overset{\mathbb{P}}{\sim} \mathcal{N}\big(\mu_z^\theta(z_t, a_t),\, \Sigma_z^\theta(z_t, a_t)\big), \\ x_t &\overset{\mathbb{P}}{\sim} \mathcal{N}\big(\mu_x^\theta(z_t, a_{t-1}),\, \Sigma_x^\theta(z_t, a_{t-1})\big), \\ r_t &\overset{\mathbb{P}}{\sim} \mathcal{N}\big(\mu_r^\theta(z_t, a_t),\, \Sigma_r^\theta(z_t, a_t)\big). \end{aligned}$$

The second is the action network φ, which depends only on the visible states (or summaries through e.g. an LSTM or GRU) via, for example,

$$a_t \overset{\mathbb{P}}{\sim} \mathcal{N}\big(\mu_a^\varphi(x_t),\, \Sigma_a^\varphi(x_t)\big), \tag{7.4}$$

and the third is the VA approximation to the posterior network ϑ, which now must account for actions and rewards via, for example,

$$z_{t+1} \overset{\mathbb{Q}}{\sim} \mathcal{N}\big(\mu_z^\vartheta(z_t, x_{1:T}, r_{1:T}, a_{1:T}),\, \Sigma_z^\vartheta(z_t, x_{1:T}, r_{1:T}, a_{1:T})\big).$$

In the above, the posterior can once again use summaries rather than entire paths of x, r, a. Optimisation proceeds in a batch and iterative manner by

1. freezing the action network φ from (7.4), running (sub-optimal) strategies, and alternatively optimising over the generative model network θ and the VA approximation network ϑ to maximise the ELBO;
2. freezing the generative model and VA approximation networks θ and ϑ, and maximising expected rewards over the action network φ using simulated paths (this replaces the Dyna-Q learning replay buffer) from the model learnt in the previous step;
3. repeating from step 1 until we get convergence.

In this manner, the setup very much resembles a GAN. It would be interesting to develop DPG outlined in Sect. 2.3 to accelerate the policy iteration part of the algorithm in step 2. Perhaps there may be a way to extend the iLQG approach outlined in Sect. 4 to the case with latent factors as well.

It is tantalising to view DMM models as discrete versions of so-called neural SDEs (see Li et al. [82, 83], Gierjatowicz et al. [49], Cuchiero et al. [33]), where the drift and volatility of a process x may be outputs of an ANN,

$$dx_t = \mu^\theta(t, x_t)\,dt + \sigma^\theta(t, x_t)\,dW_t.$$

In the case of an RDMM, however, the neural SDE is specified on the latent process z. The visible state may be viewed as realisation of a measure flow that is parametrised by an ANN, with inputs being the latent neural SDE state and samples from the measure flow being the observations. These learnt dynamics may then be used in tandem with the iLQG methodology to obtain iteratively improved closed-form correction strategies.

8 Concluding remarks

There are many interesting problems related to ML/DL in financial mathematics other than those covered in this article. I touch upon a few of these strands here in the concluding remarks, but only in passing, as it is impossible to do justice to the volume of work in such limited space.

Hedging and pricing of contingent claims are some of the original and fundamental questions in financial mathematics on which significant progress has already been achieved, including Black and Scholes [6], Merton [87], Harrison and Kreps [58], Harrison and Pliska [59], Kreps [71], Delbaen and Schachermayer [38], Bálint and Schweizer [4]. These approaches, while model-agnostic and rigorous, do require specifying a model and are not data-driven. Recently, several “model-free” approaches to hedging and pricing appear in the literature. De Spiegeleer et al. [37] use Gaussian processes (GPs) to speed up valuation and interpolate implied volatility surfaces. Ferguson and Green [41] use ANNs to approximate the mapping from model parameters (e.g. volatility and maturity) to prices for fast pricing. Ludkovski and Saporito [85] use GPs for estimating option Greeks. Buehler et al. [10] use a mean–variance criterion and parametrise trading strategies with an ANN. Fernandez Arjona and Filipović [42] use a replicating martingale approach and an ANN and polynomial basis for large portfolio pricing and risk management. Gierjatowicz et al. [49] use neural SDEs to price and learn hedging strategies simultaneously.

There have been some recent advances in connecting mean-field analysis and interacting particle systems to learning in deep neural networks; see Mei et al. [86], Rotskoff and Vanden-Eijnden [101], Sirignano and Spiliopoulos [107]. One of the key insights from [107] is that (stochastic) GD steps generate a flow of measures (for the network parameters) and this flow converges in an appropriate Skorokhod space for measure-valued processes.

The choice of optimisation engine in deep learning is vitally important in obtaining sensible results, but a topic rarely discussed (at least within the financial mathematics community). An exception is Casgrain and Kratsios [30] which poses the meta-optimisation problem of how to construct optimisation algorithms based on the pathwise performance of the optimiser.

So-called "market generators" are another area where ML models are making headway. A market generator aims to generate sample paths that reflect the realised dynamics of asset prices in a manner that resembles historical data. Obvious applications are to risk management, but they may also be applied in e.g. portfolio optimisation to generate supplemental simulated data for an optimisation engine. Some of the common tasks along this line are calibrating implied volatility surfaces and generating sensible sequences of surfaces. There are several approaches in the literature including Bayer et al. [5], Buehler et al. [11], Cuchiero et al. [33], Xu et al. [119], Ning et al. [92]. There is also fruitful activity in modelling of high-frequency limit order book evolution; see Sirignano and Cont [105], Zhang et al. [121].

Interpretability of DL models is another area that requires much more work (see e.g. Montavon et al. [89]). What interpretability even means is a contentious topic, and there is a difference between post-hoc interpretability, where a learnt DL model is "explained" (see e.g. Ribeiro et al. [99]), and incorporating interpretability directly into the model structure (see e.g. Zhou et al. [122]). One concrete approach towards post-hoc interpretability is to project the model in various regions of the state space onto low-dimensional linear models that are more easily understood. Another approach is to evaluate the impact of individual variables (the weights and biases in the affine transformation) of a learnt model; see Horel and Giesecke [60].

In all, there are a multitude of directions to discover, and my hope is that we continue to build tools that intimately tie ML techniques into our arsenal, not to supplant, but rather to supplement and bolster it. As a community, financial mathematics needs a healthy ecosystem of lines of research, and I want to make it clear that I see ML as only one (albeit important) strand.

Acknowledgements The author would like to thank Hariom Tatsat for producing Fig. 4 and Álvaro Cartea for helpful comments on earlier drafts of this article.

References

1. Al-Aradi, A., Correia, A., Naiff, D., Jardim, G., Saporito, Y.: Solving nonlinear and high-dimensional partial differential equations via deep learning. Preprint (2018). Available online at https://arxiv.org/abs/1811.08782
2. Al-Aradi, A., Jaimungal, S.: Outperformance and tracking: dynamic asset allocation for active and passive portfolio management. Appl. Math. Finance **25**, 268–294 (2018)
3. Al-Aradi, A., Jaimungal, S.: Active and passive portfolio management with latent factors. Quant. Finance **21**, 1437–1459 (2021)
4. Bálint, D.Á., Schweizer, M.: Making no-arbitrage discounting-invariant: a new FTAP beyond NFLVR and NUPBR. Swiss Finance Institute Research Paper no. 18–23 (2020). Available online at http://papers.ssrn.com/sol3/papers.cfm?abstract_id=3141770
5. Bayer, C., Horvath, B., Muguruza, A., Stemper, B., Tomas, M.: On deep calibration of (rough) stochastic volatility models. Preprint (2019). Available online at https://arxiv.org/abs/1908.08806
6. Black, F., Scholes, M.: The pricing of options and corporate liabilities. J. Polit. Econ. **81**, 637–654 (1973)
7. Bo, L., Capponi, A.: Systemic risk in interbanking networks. SIAM J. Financ. Math. **6**, 386–424 (2015)
8. Borovykh, A., Pascucci, A., La Rovere, S.: Systemic risk in a mean-field model of interbank lending with self-exciting shocks. IISE Trans. **50**, 806–819 (2018)

9. Bouveret, G., Dumitrescu, R., Tankov, P.: Technological change in water use: a mean-field game approach to optimal investment timing. Preprint (2020). Available online at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3640181
10. Buehler, H., Gonon, L., Teichmann, J., Wood, B.: Deep hedging. Quant. Finance **19**, 1271–1291 (2019)
11. Buehler, H., Horvath, B., Lyons, T., Perez Arribas, I., Wood, B.: A data-driven market simulator for small data environments. Preprint (2020). Available online at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3632431
12. Caines, P.E., Huang, M.: Graphon mean field games and the GMFG equations. In: Proceedings of 2018 IEEE Conference on Decision and Control (CDC), IEEE, pp. 4129–4134. (2018)
13. Campbell, S., Chen, Y., Shrivats, A., Jaimungal, S.: Deep learning for principal-agent mean field games. arXiv preprint 2110.01127 (2021)
14. Cardaliaguet, P., Delarue, F., Lasry, J.-M., Lions, P.-L.: The Master Equation and the Convergence Problem in Mean Field Games. Princeton University Press, Princeton (2019)
15. Cardaliaguet, P., Lehalle, C.-A.: Mean field game of controls and an application to trade crowding. Math. Financ. Econ. **12**, 335–363 (2018)
16. Carmona, R., Cooney, D., Graves, C., Laurière, M.: Stochastic graphon games: I. The static case. Preprint (2019). Available online at https://arxiv.org/abs/1911.10664
17. Carmona, R., Delarue, F.: Probabilistic Theory of Mean Field Games with Applications I. Mean Field FBSDEs, Control, and Games. Springer, Berlin (2018)
18. Carmona, R., Delarue, F.: Probabilistic Theory of Mean Field Games with Applications II. Mean Field Games with Common Noise and Master Equations. Springer, Berlin (2018)
19. Carmona, R., Fouque, J.-P., Sun, L.-H.: Mean field games and systemic risk. Commun. Math. Sci. **13**, 911–933 (2015)
20. Carmona, R., Hamidouche, K., Laurière, M., Tan, Z.: Policy optimization for linear-quadratic zero-sum mean-field type games. In: Proceedings of 2020 59th IEEE Conference on Decision and Control (CDC), IEEE, pp. 1038–1043 (2020)
21. Carmona, R., Laurière, M.: Convergence analysis of machine learning algorithms for the numerical solution of mean field control and games: II – The finite horizon case. Preprint (2019). Available online at https://arxiv.org/abs/1908.01613
22. Carmona, R., Laurière, M., Tan, Z.: Linear–quadratic mean-field reinforcement learning: convergence of policy gradient methods. Preprint (2019). Available online at https://arxiv.org/abs/1910.04295
23. Carmona, R., Laurière, M., Tan, Z.: Model-free mean-field reinforcement learning: mean-field MDP and mean-field Q-learning. Preprint (2019). Available online at https://arxiv.org/abs/1910.12802
24. Cartea, Á., Jaimungal, S.: Incorporating order-flow into optimal execution. Math. Financ. Econ. **10**, 339–364 (2016)
25. Cartea, Á., Jaimungal, S., Sánchez-Betancourt, L.: Reinforcement learning for foreign exchange trading. In: Capponi, A., Lehalle, C.-A. (eds.) Machine Learning and Data Sciences for Financial Markets: A Guide to Contemporary Practices. Cambridge University Press, Cambridge (2022). To appear
26. Cartea, Á., Perez Arribas, I., Sánchez-Betancourt, L.: Optimal execution of foreign securities: a double-execution problem with signatures and machine learning. Preprint (2020). Available online at https://ssrn.com/abstract=3562251
27. Casgrain, P., Jaimungal, S.: Mean field games with partial information for algorithmic trading. Preprint (2018). Available online at https://arxiv.org/abs/1803.04094
28. Casgrain, P., Jaimungal, S.: Trading algorithms with learning in latent alpha models. Math. Finance **29**, 735–772 (2019)
29. Casgrain, P., Jaimungal, S.: Mean-field games with differing beliefs for algorithmic trading. Math. Finance **30**, 995–1034 (2020)
30. Casgrain, P., Kratsios, A.: Optimizing optimizers: regret-optimal gradient descent algorithms. Preprint (2020). Available online at https://arxiv.org/abs/2101.00041
31. Casgrain, P., Ning, B., Jaimungal, S.: Deep Q-learning for Nash equilibria: Nash-DQN. Preprint (2019). Available online at https://arxiv.org/abs/1904.10554
32. Chauvin, Y., Rumelhart, D.E.: Backpropagation: Theory, Architectures, and Applications. Psychology Press, New York (1995)
33. Cuchiero, C., Khosrawi, W., Teichmann, J.: A generative adversarial network approach to calibration of local stochastic volatility models. Risks **8**(4), 1–31 (2020)

34. Cuchiero, C., Larsson, M., Teichmann, J.: Deep neural networks, generic universal interpolation, and controlled ODEs. SIAM J. Math. Data Sci. **2**, 901–919 (2020)
35. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. **2**, 303–314 (1989)
36. Dabérius, K., Granat, E., Karlsson, P.: Deep execution-value and policy based reinforcement learning for trading and beating market benchmarks. Preprint (2019). Available online at https://ssrn.com/abstract=3374766
37. De Spiegeleer, J., Madan, D.B., Reyners, S., Schoutens, W.: Machine learning for quantitative finance: fast derivative pricing, hedging and fitting. Quant. Finance **18**, 1635–1643 (2018)
38. Delbaen, F., Schachermayer, W.: A general version of the fundamental theorem of asset pricing. Math. Ann. **300**, 463–520 (1994)
39. E, W., Han, J., Jentzen, A.: Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. Commun. Math. Stat. **5**, 349–380 (2017)
40. Fan, J., Wang, Z., Xie, Y., Yang, Z.: A theoretical analysis of deep Q-learning. In: Bayen, A.M., et al. (eds.) Proceedings of the 2nd Conference on Learning for Dynamics and Control, PMLR, pp. 486–489 (2020)
41. Ferguson, R., Green, A.: Deeply learning derivatives. Preprint (2018). Available online at https://arxiv.org/abs/1809.02233
42. Fernandez Arjona, L., Filipović, D.: A machine learning approach to portfolio pricing and risk management for high-dimensional problems. Swiss Finance Institute Research Paper no. 20–28 (2020). Available online at https://ssrn.com/abstract=3588376
43. Féron, O., Tankov, P., Tinsi, L.: Price formation and optimal trading in intraday electricity markets with a major player. Risks **8**(4), 1–21 (2020)
44. Ferreira, T.A.: Reinforced deep Markov models with applications in automatic trading. Preprint (2020). Available online at https://arxiv.org/abs/2011.04391
45. Firoozi, D., Caines, P.E.: Mean field game ε-Nash equilibria for partially observed optimal execution problems in finance. In: Proceedings of 2016 IEEE 55th Conference on Decision and Control (CDC), IEEE, pp. 268–275 (2016)
46. Firoozi, D., Jaimungal, S.: Exploratory LQG mean field games with entropy regularization. Automatica (2020, forthcoming). Available online at https://arxiv.org/abs/2011.12946
47. Gatheral, J.: No-dynamic-arbitrage and market impact. Quant. Finance **10**, 749–759 (2010)
48. Germain, M., Pham, H., Warin, X.: Neural networks-based algorithms for stochastic control and PDEs in finance. Preprint (2021). Available online at https://arxiv.org/abs/2101.08068
49. Gierjatowicz, P., Sabate-Vidales, M., Šiška, D., Szpruch, L., Žurič, Ž.: Robust pricing and hedging via neural SDEs. Preprint (2020). Available online at https://ssrn.com/abstract=3646241
50. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Commun. ACM **63**(11), 139–144 (2020)
51. Gu, H., Guo, X., Wei, X., Xu, R.: Dynamic programming principles for mean-field controls with learning. Preprint (2019). Available online at https://arxiv.org/abs/1911.07314v5
52. Gu, H., Guo, X., Wei, X., Xu, R.: Mean-field controls with Q-learning for cooperative MARL: convergence and complexity analysis. Preprint (2020). Available online at https://arxiv.org/abs/2002.04131
53. Gu, S., Lillicrap, T., Sutskever, I., Levine, S.: Continuous deep Q-learning with model-based acceleration. In: Balcan, M.F., Weinberger, K.Q. (eds.) International Conference on Machine Learning, vol. 48, pp. 2829–2838. PMLR, New York (2016)
54. Guo, X., Hu, A., Xu, R., Zhang, J.: Learning mean-field games. In: d'Alché-Buc, F., et al. (eds.) Advances in Neural Information Processing Systems, vol. 32, pp. 4966–4976. Curran Associates, Red Hook (2019)
55. Guo, X., Xu, R., Zariphopoulou, T.: Entropy regularization for mean field games with learning. Preprint (2020). Available online at https://arxiv.org/abs/2010.00145
56. Han, J., Hu, R.: Deep fictitious play for finding Markovian Nash equilibrium in multi-agent games. In: Lu, J., Ward, R. (eds.) Proceedings of the First Mathematical and Scientific Machine Learning Conference, PMLR, vol. 107, pp. 221–245 (2020)
57. Han, J., Long, J.: Convergence of the deep BSDE method for coupled FBSDEs. Probab. Uncertain. Quant. Risk **5**, 1–33 (2020)
58. Harrison, J.M., Kreps, D.M.: Martingales and arbitrage in multiperiod securities markets. J. Econ. Theory **20**, 381–408 (1979)

59. Harrison, J.M., Pliska, S.R.: Martingales and stochastic integrals in the theory of continuous trading. Stoch. Process. Appl. **11**, 215–260 (1981)
60. Horel, E., Giesecke, K.: Towards explainable AI: significance tests for neural networks. Preprint (2019). Available online at https://arxiv.org/abs/1902.06021
61. Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Netw. **4**, 251–257 (1991)
62. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**, 359–366 (1989)
63. Hu, J., Wellman, M.P.: Nash Q-learning for general-sum stochastic games. J. Mach. Learn. Res. **4**, 1039–1069 (2003)
64. Huang, M.: Large-population LQG games involving a major player: the Nash certainty equivalence principle. SIAM J. Control Optim. **48**, 3318–3353 (2010)
65. Huang, M., Caines, P.E., Malhamé, R.P.: Large-population cost-coupled LQG problems with nonuniform agents: individual-mass behavior and decentralized ε-Nash equilibria. IEEE Trans. Autom. Control **52**, 1560–1571 (2007)
66. Huang, M., Malhamé, R.P., Caines, P.E.: Large population stochastic dynamic games: closed-loop McKean–Vlasov systems and the Nash certainty equivalence principle. Commun. Inf. Syst. **6**, 221–252 (2006)
67. Huang, X., Jaimungal, S., Nourian, M.: Mean-field game strategies for optimal execution. Appl. Math. Finance **26**, 153–185 (2019)
68. Huré, C., Pham, H., Warin, X.: Deep backward schemes for high-dimensional nonlinear PDEs. Math. Comput. **89**, 1547–1579 (2020)
69. Jaimungal, S., Pesenti, S., Wang, Y.S., Tatsat, H.: Robust risk-aware reinforcement learning. SIAM J. Financ. Math. (2021, forthcoming). https://arxiv.org/abs/2108.10403
70. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. Preprint (2014). Available online at https://arxiv.org/abs/1412.6980
71. Kreps, D.M.: Arbitrage and equilibrium in economies with infinitely many commodities. J. Math. Econ. **8**, 15–35 (1981)
72. Krishnan, R., Shalit, U., Sontag, D.: Structured inference networks for nonlinear state space models. In: Singh, S., Markovitch, S. (eds.) Proceedings of the AAAI Conference on Artificial Intelligence, pp. 2101–2109. AAAI Press, Menlo Park (2017)
73. Krishnan, R.G., Shalit, U., Sontag, D.: Deep Kalman filters. Preprint (2015). Available online at https://arxiv.org/abs/1511.05121
74. Kumar, P.: Deep recurrent Q-networks for market making. Preprint (2020). Available online at http://agi-conf.org/2020/wp-content/uploads/2020/06/AGI-20_paper_39.pdf
75. Kurutach, T., Clavera, I., Duan, Y., Tamar, A., Abbeel, P.: Model-ensemble trust-region policy optimization. Preprint (2018). Available online at https://arxiv.org/abs/1802.10592
76. Lasry, J.M., Lions, P.L.: Jeux à champ moyen. I – Le cas stationnaire. C. R. Acad. Sci. **343**, 619–625 (2006)
77. Lasry, J.M., Lions, P.L.: Jeux à champ moyen. II – Horizon fini et contrôle optimal. C. R. Acad. Sci. **343**, 679–684 (2006)
78. Lasry, J.-M., Lions, P.-L.: Mean field games. Jpn. J. Math. **2**, 229–260 (2007)
79. Lehalle, C.-A., Mouzouni, C.: A mean field game of portfolio trading and its consequences on perceived correlations. Preprint (2019). Available online at https://arxiv.org/abs/1902.09606
80. Lehalle, C.-A., Neuman, E.: Incorporating signals into optimal trading. Finance Stoch. **23**, 275–311 (2019)
81. Li, W., Todorov, E.: Iterative linear quadratic regulator design for nonlinear biological movement systems. In: Araújo, H., et al. (eds.) Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics, pp. 222–229 (2004)
82. Li, X., Wong, T.-K.L., Chen, R.T., Duvenaud, D.: Scalable gradients for stochastic differential equations. In: Chiappa, S., Calandra, R. (eds.) Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, PMLR, vol. 108, pp. 3870–3882 (2020)
83. Li, X., Wong, T.-K.L., Chen, R.T., Duvenaud, D.K.: Scalable gradients and variational inference for stochastic differential equations. In: Zhang, C., et al. (eds.) Proceedings of the 2nd Symposium on Advances in Approximate Bayesian Inference, PMLR, pp. 1–28 (2020)
84. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. Preprint (2015). Available online at https://arxiv.org/abs/1509.02971

85. Ludkovski, M., Saporito, Y.: KrigHedge: Gaussian process surrogates for delta hedging. Preprint (2020). Available online at https://arxiv.org/abs/2010.08407v3
86. Mei, S., Montanari, A., Nguyen, P.-M.: A mean field view of the landscape of two-layer neural networks. Proc. Natl. Acad. Sci. **115**(33), E7665–E7671 (2018)
87. Merton, R.C.: Theory of rational option pricing. Bell J. Econ. Manag. Sci. **4**, 141–183 (1973)
88. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G.: Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015)
89. Montavon, G., Samek, W., Müller, K.-R.: Methods for interpreting and understanding deep neural networks. Digit. Signal Process. **73**, 1–15 (2018)
90. Nevins, D.: Goals-based investing: integrating traditional and behavioral finance. J. Wealth Manag. **6**(4), 8–23 (2004)
91. Ni, C., Li, Y., Forsyth, P., Carroll, R.: Optimal asset allocation for outperforming a stochastic benchmark target. Preprint (2020). Available at SSRN 3619332. Available online at https://ssrn.com/abstract=3619332
92. Ning, B., Jaimungal, S., Zhang, X., Bergeron, M.: Arbitrage-free implied volatility surface generation with variational autoencoders. Preprint (2021). Available online at https://arxiv.org/abs/2108.04941v1
93. Ning, B., Lin, F.H.T., Jaimungal, S.: Double deep Q-learning for optimal execution. Preprint (2018). Available online at https://arxiv.org/abs/1812.06600
94. Ormerod, J.T., Wand, M.P.: Explaining variational approximations. Am. Stat. **64**(2), 140–153 (2010)
95. Pesenti, S.M., Jaimungal, S.: Portfolio optimisation within a Wasserstein ball. Preprint (2020). Available online at https://arxiv.org/abs/2012.04500
96. Pham, H.: Continuous-Time Stochastic Control and Optimization with Financial Applications. Springer, Berlin (2009)
97. Polyak, B.T.: Some methods of speeding up the convergence of iteration methods. USSR Comput. Math. Math. Phys. **4**(5), 1–17 (1964)
98. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York (2014)
99. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should I trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144. Association for Computing Machinery, New York (2016)
100. Riedmiller, M.: Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method. In: Gama, J., et al. (eds.) Machine Learning: ECML 2005, pp. 317–328. Springer, Berlin (2005)
101. Rotskoff, G.M., Vanden-Eijnden, E.: Trainability and accuracy of neural networks: an interacting particle system approach. Preprint (2018). Available online at https://arxiv.org/abs/1805.00915
102. Saporito, Y.F., Zhang, Z.: PDGM: a neural network approach to solve path-dependent partial differential equations. Preprint (2020). Available online at https://arxiv.org/abs/2003.02035
103. Shrivats, A., Firoozi, D., Jaimungal, S.: A mean-field game approach to equilibrium pricing in solar renewable energy certificate markets. Preprint (2021). Available online at https://arxiv.org/abs/2003.04938v5
104. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: Xing, E.P., Jebara, T. (eds.) Proceedings of the 31st International Conference on Machine Learning, PMLR, vol. 32, pp. 387–395 (2014)
105. Sirignano, J., Cont, R.: Universal features of price formation in financial markets: perspectives from deep learning. Quant. Finance **19**, 1449–1459 (2019)
106. Sirignano, J., Spiliopoulos, K.: DGM: a deep learning algorithm for solving partial differential equations. J. Comput. Phys. **375**, 1339–1364 (2018)
107. Sirignano, J., Spiliopoulos, K.: Mean field analysis of neural networks: a law of large numbers. SIAM J. Appl. Math. **80**, 725–752 (2020)
108. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: Dasgupta, S., McAllester, D. (eds.) International Conference on Machine Learning, PMLR, pp. 1139–1147 (2013)
109. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
110. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Solla, S., et al. (eds.) Advances in Neural Information Processing Systems, vol. 12, pp. 1057–1063. MIT Press, Cambridge (2000)

111. Todorov, E., Li, W.: A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In: Proceedings of the 2005 American Control Conference, 2005, IEEE, pp. 300–306 (2005)
112. Tsitsiklis, J.N.: Asynchronous stochastic approximation and Q-learning. Mach. Learn. **16**(3), 185–202 (1994)
113. Tukey, J.W.: The future of data analysis. Ann. Math. Stat. **33**, 1–67 (1962)
114. van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double Q-learning. In: Proceedings of Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16), pp. 2094–2100 (2016)
115. Wang, H., Zariphopoulou, T., Zhou, X.Y.: Reinforcement learning in continuous time and space: a stochastic control approach. J. Mach. Learn. Res. **21**, 1–34 (2020)
116. Wang, H., Zhou, X.Y.: Continuous-time mean–variance portfolio selection: a reinforcement learning framework. Math. Finance **30**, 1273–1308 (2020)
117. Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., Freitas, N.: Dueling network architectures for deep reinforcement learning. In: Balcan, M.F., Weinberger, K.Q. (eds.) International Conference on Machine Learning, PMLR, pp. 1995–2003 (2016)
118. Watkins, C.J., Dayan, P.: Q-learning. Mach. Learn. **8**, 279–292 (1992)
119. Xu, T., Wenliang, L.K., Munn, M., Acciaio, B.: COT-GAN: generating sequential data via causal optimal transport. Preprint (2020). Available online at https://arxiv.org/abs/2006.08571
120. Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., Wang, J.: Mean field multi-agent reinforcement learning. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, PMLR, pp. 5571–5580 (2018)
121. Zhang, Z., Zohren, S., Roberts, S.: DeepLOB: deep convolutional neural networks for limit order books. IEEE Trans. Signal Process. **67**, 3001–3012 (2019)
122. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2921–2929 (2016)