Machine learning for trading

In multiperiod trading with realistic market impact, determining the dynamic trading strategy that optimises the expected utility of final wealth can be difficult. Gordon Ritter shows that, with an appropriate choice of reward function, reinforcement learning techniques (specifically Q-learning) can successfully handle the risk-averse case

In this article, we show how machine learning can be applied to the problem of discovering and implementing dynamic trading strategies in the presence of transaction costs. Modern portfolio theory (which extends to multiperiod portfolio selection, ie, dynamic trading) teaches us that a rational risk-averse investor seeks to maximise the expected utility of final wealth, $\mathbb{E}[u(w_T)]$. Here, w_T is the wealth random variable, net of all trading costs, sampled at some future time T, and u is the investor's (concave, increasing) utility function. We answer the question of whether it is possible to train a machine-learning algorithm to behave as a rational risk-averse investor.

Notation. If $x, y \in \mathbb{R}^N$ are vectors, then let $xy \in \mathbb{R}^N$ and x/y denote the pointwise product and pointwise quotient (also vectors), while $x \cdot y = \sum_i x_i y_i \in \mathbb{R}$ denotes the scalar product. Let δ denote the first-difference operator for time series so that, for any time series $\{x_t : t = 1, \dots, T\}$, we have:

$$\delta x_t := x_t - x_{t-1}$$

The bold letters $\mathbb{E}$ and $\mathbb{V}$ denote the expectation and variance of a random variable.

Utility theory. The modern theory of risk-bearing owes most of its seminal developments to Pratt (1964) and Arrow (1971). Under this framework, the rational investor with a finite investment horizon chooses actions to maximise the expected utility of terminal wealth:

$$\text{maximise: } \mathbb{E}[u(w_T)] = \mathbb{E}\biggl[u\biggl(w_0 + \sum_{t=1}^{T} \delta w_t\biggr)\biggr] \tag{1}$$

where $T \geqslant 1$ is the finite horizon, $\delta w_t := w_t - w_{t-1}$ is the change in wealth and u denotes the utility function.

The investor cannot directly control future changes in wealth δw_t; rather, the investor makes trading decisions or portfolio-selection decisions that affect the probability distribution of future δw_t. In effect, the investor chooses which lottery to play.

The theory surrounding solutions of (1) is called multiperiod portfolio choice. As understood by Merton (1969), problems of the sort (1) fall naturally into the framework of optimal control theory. Unfortunately, for most realistic trading cost functions, the associated Hamilton-Jacobi-Bellman equations are too difficult to solve. Recent work has uncovered explicit solutions for quadratic costs (Gârleanu & Pedersen 2013), while efficient methods to deal with realistic (including non-differentiable) trading costs have been discussed by Kolm & Ritter (2015) and Boyd *et al* (2017).

In the theory of financial decision-making, a lottery is any random variable with units of wealth. According to the generalised meaning of the word 'lottery' (Pratt 1964), any investment is a lottery. Playing the lottery results in a risk, which is defined as any random increment to one's total wealth. The lottery could have a positive mean, in which case some investors would pay to play it; if it has a zero mean, however, a risk-averse investor will pay an amount called the risk premium to remove the risk from their portfolio.

The utility function $u : \mathbb{R} \to \mathbb{R}$ is a mapping from wealth, in dollars, to a real number with dimensionless units. The numerical value of u is important only insofar as it induces a preference relation, or ordering, on the space of lotteries. If the investor is risk averse, then u is concave (see Pratt (1964) and Arrow (1971) for more details).

A risk-neutral investor has a linear utility function, which implies they are concerned only with maximising expected wealth and are indifferent to risk. Most investors are not indifferent to risk, and hence maximising expected wealth is only a valid *modus operandi* in specific scenarios (eg, high-frequency trading), where the risk is controlled in some other way.

In the risk-neutral case, u is a linear function and (1) takes the much simpler form:

$$\text{maximise: } \mathbb{E}\biggl[\sum_{t=1}^{T} \delta w_t\biggr] \tag{2}$$

In this overly simplistic approximation (which, we emphasise, is not literally applicable to risk-averse investors), the problem reduces to a reinforcement learning problem.

In retrospect, it seems natural that reinforcement learning applies here. Reinforcement learning is a set of algorithms for directly learning value functions and, hence, finding approximate solutions to optimal control problems; multiperiod portfolio choice is a particular kind of optimal control.

Reinforcement learning. In reinforcement learning, agents learn how to choose actions in order to optimise a multiperiod cumulative 'reward'. We refer the reader to the wonderful book by Sutton & Barton (1998) and the survey article by Kaelbling *et al* (1996) for background and a history of reinforcement learning. If the per-period reward were identified with marginal wealth, then the problem would have the same mathematical form as (2). However, this is only the correct specification if risk is ignored.

The identification of (2) with the basic problem of reinforcement learning is the beginning of a good idea, but it needs scientifically rigorous development. Questions that need answering include the following:

- Reinforcement learning algorithms refer to an action space, a state space, a Markov decision process (MDP), a value function, etc. To which variables do these abstract concepts correspond when we are analysing a trading strategy?
- Realistic investors do not simply trade to maximise expected wealth; they are not indifferent to risk. Can reinforcement learning algorithms be modified to account for risk?
- What are the necessary mathematical assumptions on the random process driving financial markets? In particular, financial asset return distributions are widely known to have heavier tails than normal distributions, so our theory would not be much use if it required normality as an assumption.

After making precise assumptions concerning the trading process, and other assumptions concerning the utility function and the probability distributions of the underlying asset returns, we will show that multiperiod portfolio choice can be solved by reinforcement learning methods. In other words, we show that machines can learn to trade.

The trading process

Accounting for profit and loss. Suppose trading in a market with N assets occurs at discrete times $t = 0, 1, 2, \ldots, T$. Let $n_t \in \mathbb{Z}^N$ denote the holdings vector in shares at time t, so:

$$h_t := n_t p_t \in \mathbb{R}^N$$

denotes the vector of holdings in dollars, where p_t denotes the vector of midpoint prices at time t.

Assume for each t that a quantity δn_t shares are traded in the instant just before t, and no further trading occurs until the instant before $t+1$. Let:

$$v_t := \mathrm{nav}_t + \mathrm{cash}_t \quad \text{where } \mathrm{nav}_t := n_t \cdot p_t$$

denote the 'portfolio value', which we define as the net asset value in risky assets, plus cash. The profit and loss (P&L) before commissions and financing over the interval $[t, t+1)$ is given by the change in portfolio value δv_{t+1}.

For example, suppose we purchase $\delta n_t = 100$ shares of stock just before t at a per-share price of $p_t = 100$ dollars. Then, nav_t increases by 10,000, while cash_t decreases by 10,000, leaving v_t invariant. Suppose just before $t+1$ no further trades have occurred and $p_{t+1} = 105$; then, $\delta v_{t+1} = 500$, although this P&L is said to be unrealised until we trade again and move the profit into the cash term, at which point it is realised.

Now, suppose $p_t = 100$ but, due to bid-offer spread, temporary impact or some other related friction, our effective purchase price is $\tilde{p}_t = 101$. Further, suppose we continue to use the midpoint price p_t to 'mark to market', or compute the net asset value. As a result of the trade, nav_t increases by $(\delta n_t) p_t = 10{,}000$, while cash_t decreases by 10,100; this means v_t is decreased by 100, even though the reference price p_t has not changed. This difference is called slippage, and it shows up as a cost term in the cash part of v_t.

Executing the trade list results in a change in cash balance given by:

$$\delta(\mathrm{cash})_t = -\delta n_t \cdot \tilde{p}_t$$

where $\tilde{p}_t$ is our effective trade price including slippage. If the components of δn_t were all positive, then this would represent payment of a positive amount of cash; however, if the components of δn_t were negative, we would receive cash proceeds.

Hence, before financing and borrowing cost, one has:

$$\begin{aligned} \delta v_t &:= v_t - v_{t-1} = \delta(\mathrm{nav})_t + \delta(\mathrm{cash})_t \\ &= \delta n_t \cdot (p_t - \tilde{p}_t) + h_{t-1} \cdot r_t \end{aligned} \tag{3}$$

where the asset returns are $r_t := p_t / p_{t-1} - 1$. Let us define the total cost c_t inclusive of both slippage and borrowing/financing cost, as follows:

$$c_t := \mathrm{slip}_t + \mathrm{fin}_t \tag{4}$$

where:

$$\mathrm{slip}_t := \delta n_t \cdot (\tilde{p}_t - p_t) \tag{5}$$

and where fin_t denotes the commissions and financing costs incurred over the period. The commissions are proportional to δn_t, and the financing costs are convex functions of the components of n_t. The component slip_t is called the slippage cost. Our conventions are such that $\mathrm{fin}_t > 0$ always, and $\mathrm{slip}_t > 0$ with high probability, due to market impact and bid-offer spreads.

Portfolio value versus wealth. Combining (4) and (5) with (3), we have:

$$\delta v_t = h_{t-1} \cdot r_t - c_t \tag{6}$$

If we could liquidate the portfolio at the midpoint price vector p_t, then v_t would represent the total wealth at time t associated with the trading strategy under consideration. Due to slippage, it is unreasonable to expect a portfolio can be liquidated at prices p_t, which gives rise to costs of the form (5).

Concretely, $v_t = \mathrm{nav}_t + \mathrm{cash}_t$ has a cash portion and a non-cash portion. The cash portion is already in units of wealth, while the non-cash portion $\mathrm{nav}_t = n_t \cdot p_t$ could be converted to cash if a cost were paid; that cost is known as liquidation slippage:

$$\mathrm{liqslip}_t := -n_t \cdot (\tilde{p}_t - p_t)$$

Hence, it is the formula for slippage, but with $\delta n_t = -n_t$. Note that liquidation is relevant at most once per episode, meaning the liquidation slippage should be charged at most once, after the final time T.

To summarise, we may identify v_t with the wealth process w_t as long as we are willing to add a single term of the form:

$$\mathbb{E}[\mathrm{liqslip}_T] \tag{7}$$

to the multiperiod objective. If T is large and the strategy is profitable, or if the portfolio is small compared with the typical daily trading volume, then $\mathrm{liqslip}_T \ll v_T$, and (7) can be neglected without much influence on the resulting policy. For simplicity, in the following we identify v_t with total wealth w_t.

Mean-variance equivalence. The goal of recent approaches (Boyd *et al* 2017; Gârleanu & Pedersen 2013; Kolm & Ritter 2015) to multiperiod portfolio choice has been to determine the optimal deterministic policy, ie, the policy that maximises the expected utility of final wealth (1), assuming the policy is followed. For non-linear functions u, the optimal-policy problem may be difficult to attack directly. Fortunately, there are cases in which the solution is also the solution to a much easier problem: the mean-variance problem.

Let $\boldsymbol{r}$ denote the $N \times T$ random matrix of asset returns over all future periods being considered. Each column r_t of $\boldsymbol{r}$ denotes a cross-section of asset returns for a particular time period in the future.

Asset returns $\boldsymbol{r}$ drive wealth increments $\delta v_t = \delta w_t$ by (6), hence asset returns are the primary source of randomness in w_T. Expressions such as $\mathbb{E}[u(w_T)]$ refer to the expectation value with respect to $\boldsymbol{r}$.

Definition 1 The underlying asset return random variable $\boldsymbol{r}$ is said to follow a mean-variance equivalent distribution if it has a density $p(\boldsymbol{r})$ as well as first and second moments, and if, for any increasing utility function

u, there exists a constant $\kappa > 0$ such that the policy which maximised $\mathbb{E}[u(w_T)]$ is also optimal for the simpler problem:

$$\max_{\pi}\{\mathbb{E}[w_T] - (\kappa/2)\mathbb{V}[w_T]\} \tag{8}$$

Mean-variance equivalence presents a vast simplification, because while u is generally a non-linear (concave) function, one has:

$$\mathbb{E}[w_T] = w_0 + \sum_t \mathbb{E}[\delta w_t]$$

where w_0 is the initial wealth, a constant. If δw_t is statistically independent of δw_s whenever $t \neq s$, then:

$$\mathbb{V}[w_T] = \sum_t \mathbb{V}[\delta w_t] \tag{9}$$

Problem (8) then becomes:

$$\max_{\pi} \sum_t (\mathbb{E}[\delta w_t] - (\kappa/2)\mathbb{V}[\delta w_t]) \tag{10}$$

The multivariate normal is mean-variance equivalent, but the converse is false; many heavy-tailed distributions (such as the multivariate Student's t) are also mean-variance equivalent. It is not hard to show that any elliptical distribution (ie, having characteristic function $\phi(\boldsymbol{t}) = \exp(\mathrm{i}\boldsymbol{t}'\boldsymbol{\mu})\psi(\boldsymbol{t}'\boldsymbol{\Omega}\boldsymbol{t})$, where $\boldsymbol{\Omega}$ is positive definite) is mean-variance equivalent.

■ **Assumptions.** Throughout the rest of this article, we assume the multivariate distribution $p(\boldsymbol{r})$ is mean-variance equivalent. This entails that we may solve (1) by equivalently solving the (easier) problem (10).

Reinforcement learning

Many intelligent actions are deemed 'intelligent' precisely because they are optimal interactions with an environment. An algorithm plays a computer game intelligently if it can optimise the score. A robot navigates intelligently if it finds the shortest path with no collisions.

In a historical thread largely independent of the utility-based theory of portfolio selection, researchers in artificial intelligence developed models of an agent that 'learns' to interact with the agent's environment, with the eventual goal of optimising cumulative 'reward' or positive reinforcement over time.

To solve these problems, machine learning researchers developed a deep and beautiful set of theoretical results, along with corresponding algorithms and engineering solutions, under the name reinforcement learning. These developments are too numerous to list here, but many are covered in Sutton & Barton (1998) and the references therein.

Like the classic work of Merton (1969) on optimal lifetime consumption, reinforcement learning has its roots in Bellman's theory of optimal control. The main difference is that in reinforcement learning, the connection between actions and rewards (ie, the function to be optimised in optimal control) is typically unknown, and therefore must be inferred from the agent's interactions with the environment.

■ **States.** Formulating an intelligent behaviour as a reinforcement learning problem begins with identifying the state space $\mathcal{S}$. The relation with what statisticians call 'state-space models' is direct and intuitive: a Markov decision process (MDP) underlies most reinforcement learning problems.

In reinforcement learning, the environment is defined as everything outside the agent's direct control. The agent can still have arbitrarily complete knowledge of the environment, and the environment may be indirectly affected by the agent's actions. The term 'state', in reinforcement learning problems, usually refers to the state of the environment.

In trading problems, the environment should be interpreted to mean all processes generating observable data that the agent will use to make a trading decision. Let s_t denote the state of the environment at time t; the state is a data structure containing all of the information the agent will need to decide on the action. This will include the agent's current position, which is clearly an observable that is an important determinant of the next action.

At time t, the state s_t must also contain the prices p_t; beyond that, much more information may be considered. In order to know how to interact with the market microstructure and what the trading costs will be, the agent may wish to observe the bid-offer spread and liquidity of the instrument. If the decision to trade is driven by a signal – something that is supposed to be predictive of future returns – then this signal is part of the environment; the state would necessarily contain the signal.

If the process is to be Markov, then the action decision must not depend on the whole history of states; hence, the state itself must be a sufficiently rich data structure to make the optimal decision.

This prescription means the state of the environment is a fairly high-dimensional object. Fortunately, in many cases it can be simplified. With realistic trading costs, the multiperiod trading problem is already interesting and useful, even for a single asset. In cases where the cross-asset correlation can be ignored (eg, if it is being hedged externally), a multi-asset problem decouples into a system of single-asset problems. For a single asset, the state described above is not such a high-dimensional object.

■ **Actions.** In a general reinforcement learning setup, the agent observes the state and chooses an action according to some policy. This choice influences both the transition to the next state and the reward the agent receives. More precisely, a distribution $p(s', r \mid s, a)$ for the joint probability of transitioning to state $s' \in \mathcal{S}$ and receiving reward r is assumed, conditional on the previous state being s and the agent taking action a.

In portfolio choice problems, the space $\mathcal{A}$ of actions available at time t can be identified with either the space of trades δn_t or, equivalently, the space of target portfolios h_t. In general, the space of admissible actions depends on the current state, but we always denote the action space by $\mathcal{A}$; when only a subset of $\mathcal{A}$ is admissible, this will be made clear.

The action a_t leading to h_t means that:

$$\delta v_{t+1} = h_t \cdot r_{t+1} - c_{t+1}$$

and, hence, the joint distribution of all subsequent value updates $\{\delta v_{t+1}, \dots, \delta v_T\}$, as seen at time t, is determined by the action a_t.

In cases where the agent's interaction with the market microstructure is important, there will typically be more choices to make, and hence a larger action space. For example, the agent could decide which execution algorithm to use, whether to cross the spread or be passive, etc.

■ **Value functions and policies.** A policy π is a mapping from a state s to a probability distribution over actions: $\pi(a \mid s)$ is the probability of taking action a when in state s. The policy will be called deterministic if there is only one action with non-zero probability, in which case π is a mapping from the current state to the next action.

Following the notation of Sutton & Barton (1998), the sequence of rewards received after time step t is denoted $R_{t+1}, R_{t+2}, R_{t+3}, \dots$. The agent's goal is to maximise the expected cumulative reward, denoted by:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \tag{11}$$

The agent then searches for policies that maximise $\mathbb{E}[G_t]$. The sum in (11) can be either finite or infinite. The constant $\gamma \in [0, 1]$ is known as the discount rate, and it is especially useful in considering the problem with $T = \infty$, in which case γ is needed for convergence.

According to Sutton & Barton (1998), the key idea of reinforcement learning 'is the use of value functions to organize and structure the search for good policies'. The state-value function for policy π is:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

where $\mathbb{E}_\pi$ denotes the expectation under the assumption that policy π is followed. Similarly, the action-value function expresses the value of starting in state s, taking action a, and then following policy π thereafter:

$$q_\pi(s, a) := \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

Policy π is defined to be at least as good as π' if $v_\pi(s) \geqslant v_{\pi'}(s)$ for all states s. An optimal policy is defined to be one that is at least as good as any other policy. There need not be a unique optimal policy, but all optimal policies share the same optimal state-value function $v_*(s) = \max_\pi v_\pi(s)$ and optimal action-value function $q_*(s, a) = \max_\pi q_\pi(s, a)$.

The state-value function and the action-value function satisfy Bellman optimality equations:

$$v_*(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')]$$

$$q_*(s, a) = \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big]$$

where the sum over s', r denotes a sum over all states s' and all rewards r. In a continuous formulation, these sums would be replaced by integrals.

If we possess a function $q(s, a)$ that is an estimate of $q_*(s, a)$, then the greedy policy is defined as picking at time t the action a_t^* that maximises $q(s_t, a)$ over all possible a, where s_t is the state at time t. To ensure, in the limit as the number of steps increases, every action will be sampled an infinite number of times, we use an ε-greedy policy (with probability $1 - \varepsilon$, follow the greedy policy; with probability ε, uniformly sample the action space).

Given the function q_*, the greedy policy is optimal. Hence, an iterative method that converges to q_* constitutes a solution to the original problem of finding the optimal policy.

■ **Q-learning.** An important breakthrough in reinforcement learning came when Watkins (1989) suggested an iterative method that converges to the optimal action-value function q_*. The algorithm consists of the following steps. One initialises a matrix Q with one row per state and one column per action. This matrix can be the zero matrix initially, or initialised with some prior information, if available. Let S denote the current state. Repeat the following steps until a preselected convergence criterion is obtained.

■ (1) Choose action $A \in \mathcal{A}$ using a policy derived from Q (eg, the ε-greedy policy described above).

■ (2) Take action A, after which the new state of the environment is S' and we observe reward R.

■ (3) Update the value of $Q(S, A)$:

$$Q(S, A) \leftarrow Q(S, A) + \alpha\Big[R + \gamma \max_a Q(S', a) - Q(S, A)\Big] \tag{12}$$

where $\alpha \in (0, 1)$ is called the step-size parameter, which influences the rate of learning.

■ **The reward function for utility maximisation.** For a reinforcement learning approach to match (10), we need R_t to be an appropriate function of wealth increments, such that the following relation is satisfied:

$$\mathbb{E}[R_t] = \mathbb{E}[\delta w_t] - (\kappa/2)\mathbb{V}[\delta w_t]$$

One such function is:

$$R_t := \delta w_t - (\kappa/2)(\delta w_t - \hat{\mu})^2 \tag{13}$$

where $\hat{\mu}$ is an estimate of a parameter representing the mean wealth increment over one period, $\mu := \mathbb{E}[\delta w_t]$. Estimation of the parameter $\hat{\mu}$ in this context is slightly circular: $\mathbb{E}[\delta w_t]$ depends on the optimal policy, which depends on the reward function, which depends on $\hat{\mu}$.

Fortunately, there is an easy (approximate) resolution to this circularity problem. We propose that, at least initially, one use the trivial biased estimator $\hat{\mu} = 0$. This will cause us to overestimate variance in the reward function during an initial burn-in period. This overestimation will not be too large. Indeed, unless the Sharpe ratio of the strategy is very high, one has the approximation:

$$\mathbb{E}[(\delta w_t - \hat{\mu})^2] \approx \mathbb{E}[(\delta w_t)^2] \tag{14}$$

Once the value function has sufficiently converged using the approximate reward function:

$$R_t \approx \delta w_t - (\kappa/2)(\delta w_t)^2$$

one may begin to estimate $\hat{\mu}$ by the sample average. We emphasise that accurate estimation of $\hat{\mu}$ is not crucially important in obtaining a good policy, due to (14).

If we identify the reward function as (13), maximising the expected value of (11) with $\gamma = 1$ is the same as maximising the utility of final wealth:

$$\begin{aligned}\mathbb{E}[G_t] &= \mathbb{E}[R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T] \\ &= \sum_{s=t}^{T} (\mathbb{E}[\delta w_s] - (\kappa/2)\mathbb{V}[\delta w_s])\end{aligned}$$

As noted in our discussion of the trading process, δw_t can be identified with δv_t, where, as in (6):

$$\delta w_t \approx \delta v_t = h_{t-1} \cdot r_t - c_t$$

We now discuss how the reward is observed during the trading process. A microsecond before time t, the agent observes the state p_t and decides an action, which is a trade list δn_t in units of shares. The agent submits this trade list to an execution system; they can then do nothing until just before $t + 1$. The agent waits one period and observes the reward:

$$R_{t+1} \approx \delta v_{t+1} - (\kappa/2)(\delta v_{t+1})^2 \tag{15}$$

If the reinforcement algorithm is working, then the agent will learn how to maximise the cumulative reward, ie, the sum of (15), which must necessarily approximate the mean-variance form $\mathbb{E}[\delta v] - (\kappa/2)\mathbb{V}[\delta v]$.

A detailed example

In this section, we provide a proof of concept in a controlled numerical simulation that permits an approximate arbitrage, and we verify that the Q-learning agent finds and exploits this arbitrage. As mentioned above, multiperiod trading problems are mathematically interesting even when only a single asset is being considered. We consider one such problem here.

For this example, assume there exists a tradable security with a strictly positive price process $p_t > 0$. (This 'security' could itself be a portfolio of other securities, such as an exchange-traded fund or a hedged relative-value trade.) Further, suppose there is some 'equilibrium price' p_e such that $x_t = \log(p_t/p_e)$ has dynamics:

$$\mathrm{d}x_t = -\lambda x_t + \sigma\xi_t \tag{16}$$

where $\xi_t \sim N(0,1)$, and ξ_t and ξ_s are independent when $t \neq s$. This means that p_t tends to revert to its long-run equilibrium level p_e, with mean-reversion rate λ, and is a standard discretisation of the Ornstein-Uhlenbeck process. For this exercise, the parameters of the dynamics (16) were taken to be $\lambda = \log(2)/H$, where $H = 5$ is the half-life, $\sigma = 0.1$, and the equilibrium price is $p_e = 50$.

All realistic trading systems have limits that bound their behaviour. For this example, we use a reduced space of actions, in which the trade size δn_t in a single interval is limited to at most K round lots, where a 'round lot' is usually 100 shares (most institutional equity trades are in integer multiples of round lots). In addition, we assume a maximum position size of M round lots. Consequently, the space of possible trades, and also the action space, is:

$$\mathcal{A} = \text{LotSize} \cdot \{-K, -K+1, \dots, K\}$$

The action space has cardinality $|\mathcal{A}| = 2K + 1$. If we let $\mathcal{H}$ denote the possible values for the holding n_t, then, similarly, $\mathcal{H} = \{-M, -M+1, \dots, M\}$ with cardinality $|\mathcal{H}| = 2M + 1$. For the examples below, we take $K = 5$ and $M = 10$.

Another feature of real markets is the tick size, defined as a small price increment (such as US\$0.01) such that all quoted prices (ie, all bids and offers) are integer multiples of the tick size. Tick sizes exist in order to balance price priority and time priority. This is convenient for us, as we want to construct a discrete model anyway. We use TickSize $= 0.1$ for our example.

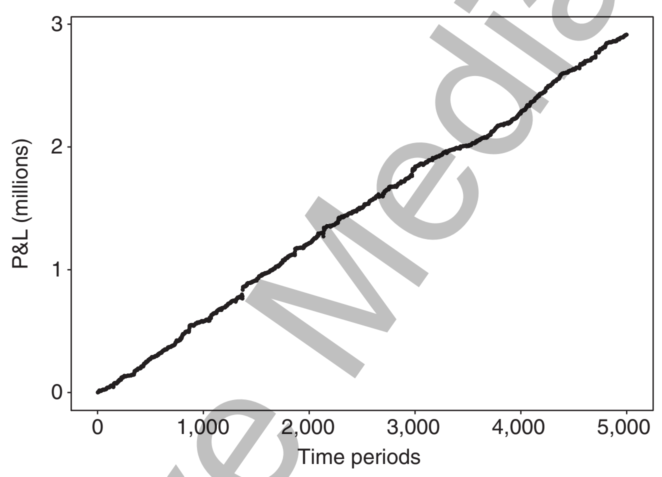
We choose boundaries of the (finite) space of possible prices so that sample paths of the process (16) exit the space with vanishingly small probability. With the parameters as above, the probability that the price path ever exits the region [0.1, 100] is small enough that no aspect of the problem depends on these bounds. Concretely, the space of possible prices is:

$$\mathcal{P} = \text{TickSize} \cdot \{1, 2, \dots, 1000\} \subset \mathbb{R}_+$$

We do not allow the agent, initially, to know anything about the dynamics. Hence, the agent does not know λ, σ or even that some dynamics of the form (16) are valid.

The agent also does not know the trading cost. We charge a spread cost of one tick size for any trade. If the bid-offer spread were equal to two ticks, then this fixed cost would correspond to the slippage incurred by an aggressive fill that crosses the spread to execute. If the spread is only one tick, then our choice is overly conservative. Hence:

1 Cumulative simulated out-of-sample P&L of trained model (over 5,000 periods)



$$\text{SpreadCost}(\delta n) = \text{TickSize} \cdot |\delta n| \tag{17}$$

We also assume there is a permanent price impact that has a linear functional form. That is, each round lot traded is assumed to move the price one tick, which leads to a dollar cost $|\delta n_t| \times \text{TickSize}/\text{LotSize}$ per share traded, giving us a total dollar cost for all shares:

$$\text{ImpactCost}(\delta n) = (\delta n)^2 \times \text{TickSize}/\text{LotSize} \tag{18}$$

The total cost is the sum of (17) and (18). Our claim is not that these are the exact cost functions for the world we live in, although the functional form does make some sense. For simplicity, we have purposely ignored the differences between temporary and permanent impact, modelling the total effect of all market impact as (18). The question is this: can an agent learn to trade with the simplest realistic interpretation of bid-offer spread and market impact? If so, then more intricate effects such as the intraday reversion of temporary impact should be studied.

As mentioned above, the state of the environment $s_t = (p_t, n_{t-1})$ will contain the security prices p_t and the agent's position, in shares, coming into the period n_{t-1}. Therefore, the state space is the Cartesian product $\mathcal{S} = \mathcal{H} \times \mathcal{P}$.

The agent then chooses an action $a_t = \delta n_t \in \mathcal{A}$, which changes the position to $n_t = n_{t-1} + \delta n_t$, and observes a P&L equal to $\delta v_t = n_t(p_{t+1} - p_t) - c_t$ and a reward $R_{t+1} = \delta v_{t+1} - 0.5\kappa(\delta v_{t+1})^2$, as in (15).

We train the Q-learner by repeatedly applying the update procedure (12). The system has various parameters that control the learning rate, discount rate, risk aversion, etc. For completeness, the parameter values used in the following example are: $\kappa = 10^{-4}$, $\gamma = 0.999$, $\alpha = 0.001$ and $\varepsilon = 0.1$. We use $n_{\text{train}} = 10^7$ training steps (each 'training step' consists of one action-value update as per (12)) and then evaluate the system on 5,000 new samples of the stochastic process.

The excellent performance out-of-sample should perhaps be expected; the assumption of an Ornstein-Uhlenbeck process implies a near arbitrage in the system. When the price is too far out of equilibrium, a trade betting it returns to equilibrium has a very small probability of loss. With our parameter settings, this is true even after costs. Thus, the existence of an arbitrage-like trading strategy in this idealised world is not surprising, and perfect mean-reverting processes such as (16) need not exist in real markets.

The surprising point is the Q-learner does not, at least initially, know there is mean-reversion in asset prices, nor does it know anything about the cost of trading. At no point does it compute estimates for the parameters λ and σ. It learns to maximise expected utility in a model-free context, ie, directly from rewards rather than indirectly (using a model).

Expected utility maximisation achieves a much higher out-of-sample Sharpe ratio than expected-profit maximisation. The distinction lies entirely in the value of κ used in (15). Choosing $\kappa = 0$ and, hence, ignoring risk is unappealing to the intuition, and indeed one may verify that this leads to a much lower Sharpe ratio. This should not be surprising. Our understanding of this principle dates back to at least 1713, when Bernoulli pointed out that a wealth-maximising investor behaves nonsensically when faced with gambling based on a martingale (see Bernoulli (1954) for a translation).

Simulation-based approaches

A major drawback of the procedure we have presented here is it requires a large number of training steps (a few million for the problem we presented). There are, of course, financial data sets with millions of time steps (eg, high-frequency data sampled once per second for several years), but a different approach is needed in other cases. Even in high-frequency examples, one may not wish to use several years' worth of data to train the model.

Fortunately, a simulation-based approach presents an attractive resolution to these issues. In other words, we propose a multistep training procedure: (1) posit a reasonably parsimonious stochastic process model for asset returns with relatively few parameters; (2) estimate the parameters of the model from market data, ensuring reasonably small confidence intervals for the parameter estimates; (3) use the model to simulate a much larger data set than the real world presents; and (4) train the reinforcement learning system on the simulated data.

For the model $dx_t = -\lambda x_t + \sigma \xi_t$, this amounts to estimating λ and σ from market data, which meets the criteria of a parsimonious model. Suppose we also have a realistic simulator of how the market microstructure will respond to various order-placement strategies. Crucially, in order to be admissible, such a simulator should be able to accurately represent the market impact caused by trading too aggressively. With these two components – a random-process model of asset returns and a good microstructure simulator – one may then run the simulation until the Q-function has converged to the optimal action-value function q_*.

The learning procedure is then only partially model free: it requires a model for asset returns, but no explicit functional form to model trading costs. The 'trading cost model' in this case is provided by the market microstructure simulator, which arguably presents a much more detailed picture than trying to distil trading costs down into a single function.

Is this procedure prone to overfitting? The answer is yes, but only if the asset-return model itself is overfitted. This procedure simply finds the optimal action-value function q_* (and, hence, the optimal policy) in the context of the model it was given. The problem of overfitting applies to the model-selection procedure used to produce the asset return model, and not directly to the reinforcement learning procedure. In the proposed training procedure, steps (1) and (2) are prone to overfitting, while steps (3) and (4) are simply a way to converge to the optimal policy in the context of the chosen model.

Conclusions

Our main contribution is to show how to handle risk-averse expected-utility maximisation using reinforcement learning. This differs substantially from approaches such as that of Gârleanu & Pedersen (2013). To use their approach requires three models: a model of expected returns, a risk model that forecasts the variance and a (pre-trade) transaction cost model. By contrast, our method can, in principle, be applied without directly estimating any of these three models, or by estimating only the expected-return model. ■

Gordon Ritter is a senior portfolio manager at GSA Capital Partners, and adjunct professor at Courant Institute at NYU, Baruch College (CUNY) and Department of Statistics Rutgers University. Email: ritter@post.harvard.edu.

REFERENCES

Arrow KJ, 1971
Essays in the Theory of Riskbearing
Markham, Chicago, IL

Bernoulli D, 1954
Exposition of a new theory on the measurement of risk
Econometrica 22(1), pages 23–36

Boyd S, E Busseti, S Diamond, RN Kahn, K Koh, P Nystrup and J Speth, 2017
Multi-period trading via convex optimization
Preprint, arXiv:1705.00109

Gârleanu N and LH Pedersen, 2013
Dynamic trading with predictable returns and transaction costs
Journal of Finance 68(6), pages 2309–2340

Kaelbling LP, ML Littman and AW Moore, 1996
Reinforcement learning: a survey
Journal of Artificial Intelligence Research 4, pages 237–285

Kolm PN and G Ritter, 2015
Multiperiod portfolio selection and Bayesian dynamic models
Risk March, pages 50–54

Merton RC, 1969
Lifetime portfolio selection under uncertainty: the continuous-time case
Review of Economics and Statistics 51(3), pages 247–257

Pratt JW, 1964
Risk aversion in the small and in the large
Econometrica 32(1/2), pages 122–136

Sutton RS and AG Barto, 1998
Reinforcement Learning: An Introduction, volume 1
MIT Press, Cambridge, MA

Watkins CJCH, 1989
Learning from Delayed Rewards
PhD thesis, King's College, available at http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf