MACHINE LEARNING FOR TRADING

GORDON RITTER*

Courant Institute of Mathematical Sciences
New York University
251 Mercer St., New York, NY 10012

Abstract. In multi-period trading with realistic market impact, determining the dynamic trading strategy that optimizes expected utility of final wealth is a hard problem. In this paper we show that, with an appropriate choice of the reward function, reinforcement learning techniques (specifically, Q-learning) can successfully handle the risk-averse case. We provide a proof of concept in the form of a simulated market which permits a statistical arbitrage even with trading costs. The Q-learning agent finds and exploits this arbitrage.

1. Introduction

In this note, 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 multi-period portfolio selection, ie. dynamic trading) teaches us that a rational risk-averse investor seeks to maximize 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's possible to train a machine-learning algorithm to behave as a rational risk-averse investor.

1.1. **Notation.** Let $\mathbb{R}$ denote the real numbers. 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 for any time series

Date: August 11, 2017.
Key words and phrases. Finance; Investment analysis; Machine learning; Portfolio optimization.

* Corresponding author. *E-mail address:* `ritter@post.harvard.edu`.

Electronic copy available at: https://ssrn.com/abstract=3015609

$\{x_t : t = 1, \dots, T\}$, we have

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

We will never use the letter δ in any other way. The bold letters $\mathbb{E}$ and $\mathbb{V}$ denote the expectation and variance of a random variable.

1.2. **Utility theory.** The modern theory of risk-bearing owes most of its seminal developments to Pratt (1964) and Arrow (1971), and is summarized in many standard textbooks (Ingersoll, 1987). Under their framework, the rational investor with a finite investment horizon chooses actions to maximize the expected utility of terminal wealth:

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

where $T \geq 1$ 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 which 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 already understood by Merton (1969), problems of the sort (1) fall naturally under 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 and Pedersen, 2013), and efficient methods to deal with realistic (including non-differentiable) trading costs were discussed by Kolm and Ritter (2015), and Boyd et al. (2017).

In the theory of financial decision-making a *lottery* is any random variable with units of wealth. In the generalized meaning of the word 'lottery" due to Pratt (1964), any investment is a lottery. Playing the lottery results in a *risk*, 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, whereas if it has a zero mean then any risk-averse investor would 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 units of 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. We refer to Pratt (1964) and Ingersoll (1987) for more details.

A risk-neutral investor has a linear utility function, which implies they are concerned only with maximizing expected wealth, and they are indifferent to risk. Of course, most investors (at least those who stay in business for very long) are not indifferent to risk, and hence maximizing expected wealth is only a valid *modus operandi* in very limited kinds of scenarios (eg. high-frequency trading) where the risk is controlled in some other way.

The concept of a risk-neutral investor is often useful as a theoretical tool. In the risk-neutral case, then u is a linear function and (1) takes the much simpler form

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

In this overly-simplistic approximation (which, we emphasize, is not literally applicable to risk-averse investors), one may readily observe that the problem reduces to a reinforcement learning problem.

In retrospect we shouldn't be too surprised that reinforcement learning applies here. The theory of multiperiod portfolio selection and reinforcement learning are both closely related to the Bellman (1957) theory of optimal control. Reinforcement learning is a set of algorithms for directly learning value functions and hence finding approximate solutions to optimal control problems, and multiperiod portfolio choice is a particular kind of optimal control problem.

1.3. **Reinforcement learning.**

In reinforcement learning, agents learn how to choose actions in order to optimize a multi-period cumulative "reward" which has the same mathematical form as (2) if marginal wealth is the reward. We refer to the wonderful book by Sutton and Barto (1998) and the survey article by Kaelbling, Littman, and Moore (1996) for details.

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

(1) Reinforcement learning algorithms refer to an action space, a state space, a Markov decision process (MDP), value function, etc. To which variables do these abstract concepts correspond when analyzing a trading strategy?

(2) Realistic investors do not simply trade to maximize expected wealth; they are not indifferent to risk. Can reinforcement learning algorithms be modified to account for risk?

(3) 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 the normal distribution, so our theory wouldn't be much use if it required normality as an assumption.

In due course, 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 eventually show that multiperiod portfolio choice can be solved by reinforcement learning methods. That is, we show that machines *can* learn to trade.

2. The trading process

2.1. **Accounting for profit and loss.** Suppose that 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 that

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

denotes the vector of holdings in dollars.

Assume for each t, a quantity δn_t shares are traded in the instant just before t, and no further trading occurs until the instant before $t+1$. We shall see later that as long as we are careful in our accounting, and use prices adjusted for slippage, then the precise times within $[t, t+1)$ when the trade is executed are not important.

Let

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

denote the "portfolio value," which we define to be net asset value in risky assets, plus cash. The *profit and loss (PL)* 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 that just before $t+1$, no further trades have occurred and $p_{t+1} = 105$; then $\delta v_{t+1} = 500$, although this PL is said to be *unrealized* until we trade again and move the profit into the cash term, at which point it is *realized*.

Now suppose that $p_t = 100$ but due to bid-offer spread, temporary impact, or other related frictions our effective purchase price was $\tilde{p}_t = 101$. Suppose further that we continue to use the midpoint price p_t to "mark to market," ie. to compute the net asset value. Then as a result of the trade, nav_t increases by $(\delta n_t)p_t = 10,000$ while cash_t decreases by 10,100, which means that v_t is decreased by 100 even though the reference price p_t has not changed. This difference is called *slippage*; 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

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

where $\tilde{p}_t$ continues to denote 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, whereas if the components of δn_t were negative we receive cash proceeds.

Hence before financing and borrow cost, one has

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

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 borrow/financing cost, as follows:

$$c_t := \text{slip}_t + \text{fin}_t, \quad \text{where} \tag{4}$$

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

where fin_t denotes the commissions and financing costs incurred over the period; commissions are proportional to δn_t and 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 $\text{fin}_t > 0$ always, and $\text{slip}_t > 0$ with high probability due to market impact and bid-offer spreads.

2.2. **Portfolio value versus wealth.** Combining (4)–(5) with (3) we have finally

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

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 to the trading strategy under consideration. As discussed, due to slippage it is unreasonable to expect that a portfolio can be liquidated at prices p_t, which gives rise to costs of the form (5).

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

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

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 summarize, 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

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

at the end of the multi-period objective. If T is large and the strategy is profitable, or if the portfolio is small compared to the typical daily trading volume, then $\text{liqslip}_T \ll v_T$ and (7) can be neglected without much influence on the resulting policy. In what follows, for simplicity we will continue to identify v_t with total wealth w_t.

2.3. **Mean-variance equivalence.**

The goal of recent approaches (Gârleanu and Pedersen, 2013; Kolm and Ritter, 2015; Boyd et al., 2017) to multiperiod portfolio choice is to determine the optimal deterministic policy, i.e. the policy which maximizes the expected utility of final wealth, (1), assuming the policy is followed. For complicated nonlinear 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 $\mathbf{r}$ denote the $N \times T$ random matrix of asset returns over all future periods being considered. Each column r_t of $\mathbf{r}$ denotes a cross-section of asset returns for a particular time period in the future. Importantly, we assume r_t is statistically independent of r_s whenever $t \neq s$. Sometimes, it is more convenient to represent $\mathbf{r}$ as an NT-dimensional column vector obtained by stacking the columns; then, summations such as (9) can be written in vectorized form.

Asset returns $\mathbf{r}$ drive wealth increments $\delta v_t = \delta w_t$ by (6), and hence asset returns are the primary source of randomness in the random variable w_T.

In the following, expressions such as $\mathbb{E}[u(w_T)]$ refer to the expectation value with respect to $\mathbf{r}$.

Definition 1. The underlying asset return random variable $\mathbf{r}$ is said to follow a *mean-variance equivalent distribution* if it has a density $p(\mathbf{r})$, has first and second moments, and for any increasing utility function u, there exists a constant $\kappa > 0$ such that the policy which maximized $\mathbb{E}[u(w_T)]$ 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 nonlinear (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 from δw_s whenever $t \neq s$, then one can also say

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

Problem (8) then becomes:

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

It is easier to solve (10) than (1), so it's preferable to model asset returns with mean-variance equivalent distributions, which means we need to characterize mean-variance equivalent distributions. Tobin (1958) showed that the multivariate normal distribution is mean-variance equivalent, but the converse is false; many distributions, including heavy-tailed distributions such as the multivariate Student-t, are also mean-variance equivalent. In fact, using mathematical results of Chamberlain (1983) and following arguments in Ingersoll (1987), one may show that all elliptical distributions are mean-variance equivalent.

Assumptions. Throughout the rest of the paper, we assume that the multivariate distribution $p(\mathbf{r})$ is mean-variance equivalent. This entails, as a consequence, that we may solve (1) by equivalently solving the (easier) problem (10).

3. Reinforcement Learning

Many intelligent actions are deemed "intelligent" precisely because they are optimal interactions with an environment; for example, an algorithm plays a computer game intelligently if it can optimize the score. A robot navigates intelligently if it finds a 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 which "learns" to interact with the agent's environment, with the eventual goal of optimizing cumulative "reward" or positive reinforcement over time.

To solve these problems, machine learning researchers developed a deep and beautiful set of theoretical results and corresponding algorithms and engineering solutions, under the name *reinforcement learning.* These developments are too numerous to list here, but most of them are covered in the book by Sutton and Barto (1998) and references therein.

Like the classic work of Merton (1969) on optimal lifetime consumption, reinforcement learning has its roots in the pioneering work of Bellman (1957) on optimal control. The main difference is that in reinforcement learning, typically the connection between actions and rewards (ie. the function to be optimized in optimal control!) is unknown, and must be inferred from the agent's interactions with the environment.

3.1. **States.** Formulating an intelligent behavior as a reinforcement learning problem begins with identification of the state space $\mathcal{S}$. The identification with what statisticians call "state-space models" is direct and intuitive: underlying every reinforcement learning problem is a Markov Decision Process (MDP).

In reinforcement learning, the *environment* is defined to be everything outside the agent's *direct* control. The agent can still have arbitrarily complete knowledge about 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. We 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 in order to decide upon 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, but beyond that, there is much more information that 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 which is supposed to be predictive of future returns – then that 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, which means that the state itself must be a sufficiently rich data structure to make the optimal decision.

This prescription means that 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) then 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.

3.2. **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, as well as the reward the agent receives. More precisely, there is assumed to be a distribution $p(s', r \mid s, a)$ for the joint probability of transitioning to state $s' \in \mathcal{S}$ and receiving reward r, 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 $\mathcal{A}$, and 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 then 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.

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

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

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

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

According to Sutton and Barto (1998), "the key idea of reinforcement learning generally, is the use of value functions to organize and structure the search for good policies." The *state-value function* for policy π is

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

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:

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

Policy π is defined to be at least as good as π' if $v_\pi(s) \geq v_{\pi'}(s)$ for all states s. An *optimal policy* is defined to be one which 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 action-value function satisfy Bellman optimality equations

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

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 $q(s, a)$ which is an estimate of $q_*(s, a)$, then the *greedy policy* is defined as picking at time t the action a_t^* which maximizes $q(s_t, a)$ over all possible a, where s_t is the state at time t. In practice we want to ensure that, in the limit as the number of steps increases, every action will be sampled an infinite number of times. To ensure this we use an *ϵ-greedy policy*: with probability $1 - \epsilon$ follow the greedy policy, while with probability ϵ select randomly from amongst all available actions with equal probability.

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

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

1. Choose action $A \in \mathcal{A}$ using a policy derived from Q (for example, 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 $Q(S, A)$:

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

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

3.5. **The reward function for utility maximization.** How does multiperiod portfolio selection fit into this framework? How should we define the "rewards" to use in (10)? 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:

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

One such function is,

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

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 a fairly easy (approximate) resolution to this circularity problem. We propose that, at least initially, one use the trivial biased estimator $\hat{\mu} = 0$. This will have the effect that we overestimate variance in the reward function during an initial burn-in period. This over-estimation 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

$$R_t \approx \delta w_t - \frac{\kappa}{2}(\delta w_t)^2,$$

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

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

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

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 then can do nothing until just before $t+1$, so the agent waits one period and observes the reward

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

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

4. A Detailed Example

In this section, we provide a proof of concept in a controlled numerical simulation which 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 that 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 ETF or a hedged relative-value trade.) Further suppose that there is some "equilibrium price" p_e such that $x_t = \log(p_t/p_e)$ has dynamics

$$dx_t = -\lambda x_t + \sigma\,\xi_t \tag{16}$$

where $\xi_t \sim N(0,1)$ and ξ_t, 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 discretization 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 some limits which bound their behavior. 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). Also 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 $|\mathcal{A}| = 2K+1$. Letting $\mathcal{H}$ 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 USD 0.01) such that all quoted prices (i.e. 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 since 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 which crosses the spread to execute. If the spread is only one tick, then our choice is overly conservative. Hence

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

We also assume that there is permanent price impact which has a linear functional form: each round lot traded is assumed to move the price one tick, hence leading to a dollar cost $|\delta n_t| \times \text{TickSize}/\text{LotSize}$ per share traded, for 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, modeling the total effect of all market impact as (18). The question is whether an agent can learn to trade with the simplest realistic interpretation of bid/offer spread and market impact. If this works, then more intricate effects such as the intraday reversion of temporary impact should be studied in a following paper.

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 profit/loss 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 eq. (15).

We train the Q-learner by repeatedly applying the update procedure involving (12). The system has various parameters which control the learning rate, discount rate, risk-aversion, etc. For completeness, the parameter values used in the following example were: $\kappa = 10^{-4}$, $\gamma = 0.999$, $\alpha = 0.001$, $\epsilon = 0.1$. We use $n_{train} = 10^7$ 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 training is quite fast; with a custom Java 8 implementation running on a Core i5 macbook, the learner completes one million iterations of (12) per second. Indeed, in real-world scenarios we will struggle to have enough data for the training time to be any sort of bottleneck.

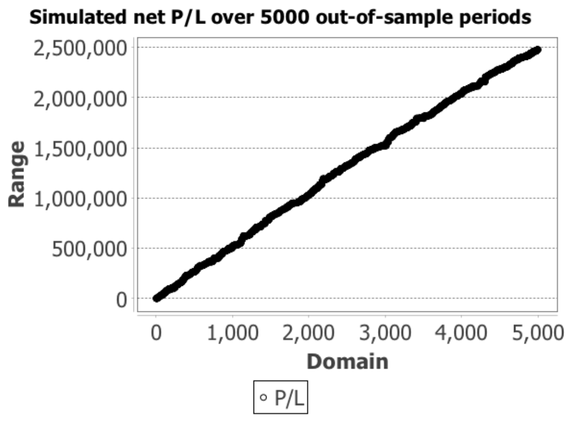

FIGURE 4.1. Cumulative simulated out-of-sample P/L of trained model.

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 that it returns to equilibrium has a very small probability of a loss. With our parameter settings, even after costs this is true. Hence the *existence* of an arbitrage-like trading strategy in this idealized world is not surprising, and perfect mean-reverting processes such as (16) need not exist in real markets.

Hence the interesting point isn't the existence of a near-arbitrage in an idealized market with frictions; rather, the point is that the Q-learner does not, at least initially, know that 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, σ. It learns to maximize expected utility in a model-free context, ie. directly from rewards rather than indirectly (using a model). This is loosely analogous to the Atari-game playing agents of Mnih et al. (2013) which learn to play without knowing the structure and rules of the game (only optimizing the score).

The other interesting point is that expected utility maximization achieves a much higher out-of-sample Sharpe ratio than expected-profit maximization. The distinction lies entirely in the value of κ used in (15). Choosing $\kappa = 0$ corresponds to the reward function of a risk-neutral agent – quite literally, an agent who doesn't care about risk. This is unappealing to the intuition, and indeed one may verify that this leads to a much lower Sharpe ratio. This should not be surprising. Understanding of this principle dates back at least to 1713 when Bernoulli pointed out that a wealth-maximizing investor behaves nonsensically when faced with gambling based on a martingale (see Bernoulli (1954) for a translation).

5. Simulation-Based Approaches

A major drawback of the procedure we have presented here is that it requires a large number of training steps (a few million, on the problem we presented). There are, of course, financial data sets with millions of time-steps (e.g. high-frequency data sampled once per second for several years), but in other cases, a different approach is needed. 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 multi-step 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, 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 distill 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 overfit. 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 that was used to produce the asset return model, and not directly to the reinforcement learning procedure. In the procedure above, steps 1-2 are prone to overitting, while steps 3-4 are simply a way to converge to the optimal policy in the context of the chosen model.

6. Conclusions

According to neoclassical finance theory, no investor should hold a portfolio (or follow a dynamic trading strategy) that does not maximize expected utility of final wealth, $\mathbb{E}[u(w_T)]$. The concave utility function u expresses the investor's *risk aversion*, or preference for a less risky portfolio over a more risky one with the same expected return. Only a risk-neutral investor would maximize $\mathbb{E}[w_T]$. The main contribution of this paper is that we show how to handle the risk-averse case in a model-free way using Q-learning. We provide a proof of concept in a controlled numerical simulation which permits an approximate arbitrage, and we verify that the Q-learning agent finds and exploits this arbitrage.

It is instructive to consider how this differs from approaches such as Gârleanu and Pedersen (2013); to use their approach in practice requires three models: a model of expected returns, a risk model which forecasts the variance, and a (pre-trade) transaction cost model. The methods of Gârleanu and Pedersen (2013) provide an explicit solution only when the cost model is quadratic. By contrast, the methods of the present paper can, in principle, be applied without directly estimating any of these three models, or they can be applied in cases where one has an asset return model, but one wishes to use machine learning techniques to infer the cost function and the optimal strategy.

References

Arrow, Kenneth J (1971). "Essays in the theory of risk-bearing".

Bellman, Richard (1957). *Dynamic Programming.*

Bernoulli, Daniel (1954). "Exposition of a new theory on the measurement of risk". *Econometrica: Journal of the Econometric Society*, pp. 23–36.

Boyd, Stephen et al. (2017). "Multi-Period Trading via Convex Optimization". *arXiv preprint arXiv:1705.00109.*

Chamberlain, Gary (1983). "A characterization of the distributions that imply mean–variance utility functions". *Journal of Economic Theory* 29.1, pp. 185–201.

Gârleanu, Nicolae and Lasse Heje Pedersen (2013). “Dynamic trading with predictable returns and transaction costs”. *The Journal of Finance* 68.6, pp. 2309–2340.

Ingersoll, Jonathan E (1987). *Theory of financial decision making.* Vol. 3. Rowman & Littlefield.

Kaelbling, Leslie Pack, Michael L Littman, and Andrew W Moore (1996). “Reinforcement learning: A survey”. *Journal of artificial intelligence research* 4, pp. 237–285.

Kolm, Petter N and Gordon Ritter (2015). “Multiperiod Portfolio Selection and Bayesian Dynamic Models”. *Risk.*

Merton, Robert C (1969). “Lifetime portfolio selection under uncertainty: The continuous-time case”. *The review of Economics and Statistics*, pp. 247–257.

Mnih, Volodymyr et al. (2013). “Playing atari with deep reinforcement learning”. *arXiv preprint arXiv:1312.5602.*

Pratt, John W (1964). “Risk aversion in the small and in the large”. *Econometrica: Journal of the Econometric Society*, pp. 122–136.

Sutton, Richard S and Andrew G Barto (1998). *Reinforcement learning: An introduction.* Vol. 1. 1. MIT press Cambridge.

Tobin, James (1958). “Liquidity preference as behavior towards risk”. *The review of economic studies* 25.2, pp. 65–86.

Watkins, Christopher JCH (1989). “Q-learning”. *PhD Thesis.*