The Usefulness of Reinforcement Learning in Finance

Gordon Ritter

November 14, 2018

New York University — Courant Institute of Mathematics and Tandon School
Rutgers University — Department of Statistics and FSRM program
Baruch College — Department of Mathematics and MFE program

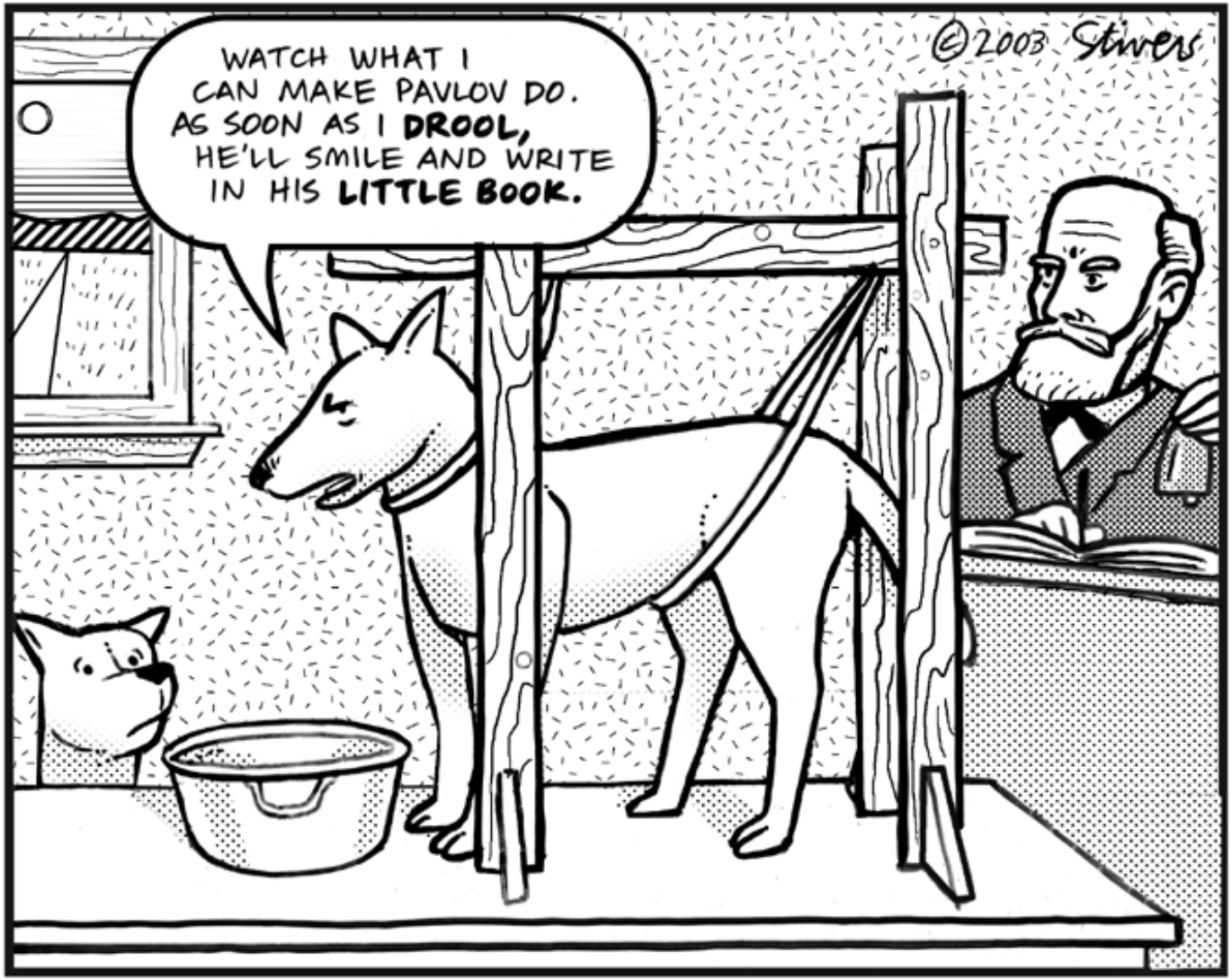
https://cims.nyu.edu/~ritter/
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3281235

- We stand at a crossroads.
- In the last decade, Artificial Intelligence has gone from a sci-fi fantasy to touching almost every aspect of our lives.
- In many tasks requiring complex multi-period planning in the presence of uncertainty (e.g. backgammon, driving, chess, Jeopardy, Atari, the ancient Chinese game of Go, etc.) agents are superior to the top humans.
- In all of the examples just mentioned, the agents were trained using a machine learning technique known as *reinforcement learning*.

What is Intelligence?

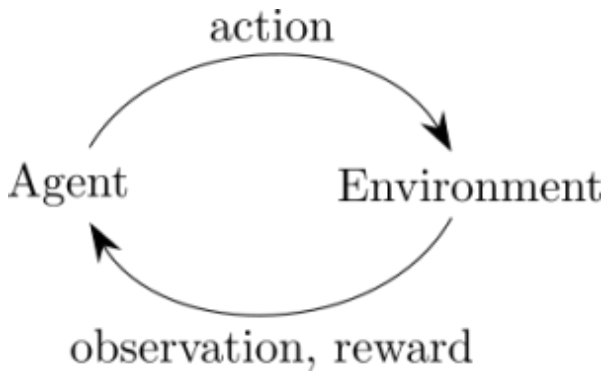
- Many intelligent actions are deemed "intelligent" precisely because they are optimal interactions with an environment.
- An algorithm plays a computer game intelligently if it can optimize the score.
- A robot navigates intelligently if it finds a shortest path with no collisions: minimizing a function which entails path length with a large negative penalty for collision.
- *Learning* is learning how to choose your actions wisely to optimize your interaction with your environment, in such a way to maximize rewards received over time.
- A gazelle is born not knowing how to walk. Its brain learns how to send signals to its leg muscles so as to optimize interactions with its environment (standing, walking, running).

- In many of these examples, the learning happens through experience, in a process roughly described as "trial and error."



General Framework for Decision Making

- An interacting system: agent interacts with environment.
- The "environment" is the part of the system outside of the agent's *direct* control.
- At each time step t, the agent observes the current state of the environment $s_t \in \mathcal{S}$, and chooses an action $a_t \in \mathcal{A}$.
- This choice influences both the transition to the next state, as well as the reward the agent receives.



- The agent's goal is to maximize the expected cumulative reward, denoted by

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \tag{1.1}$$

  where γ is necessary for the infinite sum to be defined.
- A policy π is an algorithm for choosing the next action, based on the state you are in.
- Reinforcement learning is the search for policies which maximize

$$\mathbb{E}[G_t] = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots]$$

- 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."
- Assuming I start in state s, take action a, and then follow some fixed policy π from then on, what is my expected cumulative reward over time? The answer is the *action-value function*:

$$q_\pi(s, a) := \mathbb{E}[G_t] \text{ starting from } s, \text{ taking } a, \text{ then following } \pi$$

- If we knew the q-function corresponding to the optimal policy, say q_*, we would know the optimal policy itself:

$$\text{choose } a \in \mathcal{A} \text{ to maximize } q_*(s_t, a)$$

  This is called following the *greedy policy*.
- Hence we can reduce the problem to finding q_*, or producing a sequence of iterates that converges to q_*.

Worthwhile Questions

- Can an artificial intelligence discover an *optimal* dynamic trading strategy (with transaction costs), without being told what kind of strategy to look for?

  In other words, what are the various financial analogues of AlphaGo Zero, a system which learned to play with "zero" human guidance, merely being told the rules of the game and playing against a simulator?
- What does the term "optimal" mean exactly? Is it subjective or can we quantify the kind of strategy that any rational decision-maker would employ?

- In finance, *optimal* means that the strategy optimizes expected utility of final wealth, and final wealth is the sum of a number of wealth increments over shorter time periods:

$$\text{maximize: } \mathbb{E}[u(w_T)] = \mathbb{E}[u(w_0 + \sum_{t=1}^{T} \delta w_t)] \tag{1.2}$$

where $\delta w_t := w_t - w_{t-1}$.

Utility theory

- In 1947, John von Neumann and Oskar Morgenstern proved that if a decision-maker
  (a) is faced with risky (probabilistic) outcomes of different choices, and
  (b) has preferences satisfying four axioms of "rational behavior"

  then that decision-maker will behave as if they are maximizing the expected value of some function, u, called the utility function, defined over the potential outcomes.
- When applied to trading of financial assets, the outcomes are typically various levels of wealth w_T at some future time T, and rational agents maximize

$$\mathbb{E}[u(w_T)]$$

  (not $\mathbb{E}[w_T]$, since this leads to paradoxes, and ignoring risk is ill-advised).

Under certain assumptions, maximizing $\mathbb{E}[u(w_T)]$ is equivalent to maximizing a mean-variance form of that problem. Let us clarify those assumptions.

Discreteness Trading occurs at discrete times $t = 1, \ldots, T$ and final wealth is given by

$$w_T = w_0 + \sum_{t=1}^{T} \delta w_t, \quad \delta w_t := w_t - w_{t-1}.$$

Portfolios There exist a set of portfolios $h_0, \ldots, h_{T-1}$ known at $t = 0$ such that $\delta w_t = h_{t-1} \cdot \boldsymbol{r}_t$ where $\boldsymbol{r}_t$ is the random vector of asset returns over the period $[t-1, t]$.

Independence If $t \neq s$, then $\boldsymbol{r}_t$ and $\boldsymbol{r}_s$ are independent random vectors.

Ellipticality For each t, the multivariate distribution of $\boldsymbol{r}_t$ is elliptical (iso-probability contours are multi-dimensional ellipses).

Utility The utility function is increasing, concave, and has continuous derivatives up to second order.

- We will make the above assumptions in this talk.
- Under these assumptions, there exists some constant $\kappa > 0$, which depends on initial wealth w_0 and the investor's utility function, such that maximizing $\mathbb{E}[u(w_T)]$ is equivalent to maximizing

$$\mathbb{E}[w_T] - \frac{1}{2}\,\kappa\,\mathbb{V}[w_T]$$

- So we focus on

$$\text{maximize}: \quad \left\{ \mathbb{E}[w_T] - \frac{\kappa}{2} \mathbb{V}[w_T] \right\} \tag{1.3}$$

- Suppose we could invent some definition of "reward" R_t so that

$$\mathbb{E}[w_T] - \frac{\kappa}{2} \mathbb{V}[w_T] \approx \sum_{t=1}^{T} R_t \tag{1.4}$$

  Then (1.3) looks like a "cumulative reward over time" problem.

- Reinforcement learning is the search for policies which maximize

$$\mathbb{E}[G_t] = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots]$$

  which by (1.4) would then maximize expected utility as long as $\gamma \approx 1$.

- Consider the reward function

$$R_t := \delta w_t - \frac{\kappa}{2}(\delta w_t - \hat{\mu})^2 \tag{1.5}$$

where $\hat{\mu}$ is an estimate of a parameter representing the mean wealth increment over one period, $\mu := \mathbb{E}[\delta w_t]$.

- Then

$$\frac{1}{T}\sum_{t=1}^{T} R_t = \underbrace{\frac{1}{T}\sum_{t=1}^{T} \delta w_t}_{\to \mathbb{E}[\delta w_t]} - \frac{\kappa}{2}\underbrace{\frac{1}{T}\sum_{t=1}^{T}(\delta w_t - \hat{\mu})^2}_{\to \mathbb{V}[\delta w_t]}$$

and for large T, the two terms on the right hand side approach the sample mean and the sample variance, respectively.

- Thus with this one special choice of the reward function (1.5), if the agent learns to maximize cumulative reward, it should also approximately maximize the mean-variance form of utility.

In trading problems, what are the state space, action space, and reward?

- The state variable s_t is a data structure which, simply put, must contain everything the agent needs to make a trading decision, and nothing else.
- Variables that are good candidates to include in the state:

1. the current position or holding in the asset
2. the values of any signals which are believed to be predictive
3. the current state of the market, including current price and any relevant microstructure / limit-order book details
4. If options are involved, how long until they expire?

- In trading problems, the most obvious choice for an action is the number of shares to trade, δn_t. This choice identifies the action space $\mathcal{A} \subset \mathbb{Z}$.
- If 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, target participation rate, etc.
- If one of the assets is an option, there may be additional actions available, such as early exercise.

The Ornstein-Uhlenbeck Process

- 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

$$dx_t = -\lambda x_t + \sigma\, \xi_t \tag{2.1}$$

  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.
- These assumptions imply something similar to an arbitrage! Positions taken in the appropriate direction while very far from equilibrium have very small probability of loss and extremely asymmetric loss-gain profiles.

- 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 (2.1) 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

$$\text{SpreadCost}(\delta n) = \text{TickSize} \cdot |\delta n| \tag{2.2}$$

- 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 $|\delta n_t| \times \text{TickSize/LotSize}$ per share traded, for a total dollar cost for all shares

$$\text{ImpactCost}(\delta n) = (\delta n)^2 \times \text{TickSize/LotSize}. \tag{2.3}$$

- The total cost is taken to be

$$\text{cost}(\delta n) = \text{multiplier} \times (\text{SpreadCost}(\delta n) + \text{ImpactCost}(\delta n)) \tag{2.4}$$

  This functional form matches Almgren–Chriss, and some other accounts in the literature.
- The multiplier allows us to quickly and easily test the reinforcement learning agent in regimes of more or less liquidity.

- 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}.

- We first train a tabular Q-learner with $n_{train} = 10^7$ training steps, and then evaluate the system on 5,000 new samples of the stochastic process.

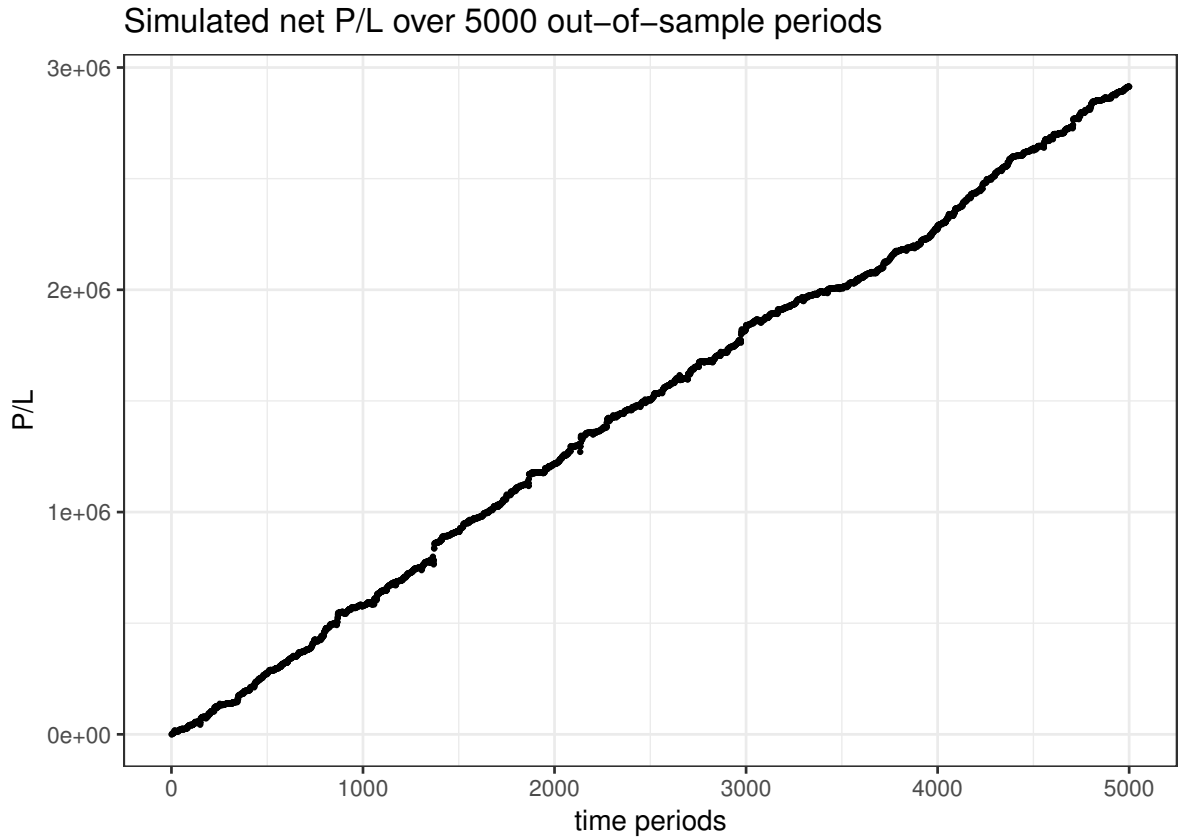


Figure: Cumulative simulated out-of-sample P/L of trained model.

- The initial results are encouraging. We constructed a system wherein we know there is an arbitrage, analogous to a game where it's actually possible to win.
- The machine then learns to play this game and learns a reasonable strategy.
- But how good is its value function, really? Let's take a look...

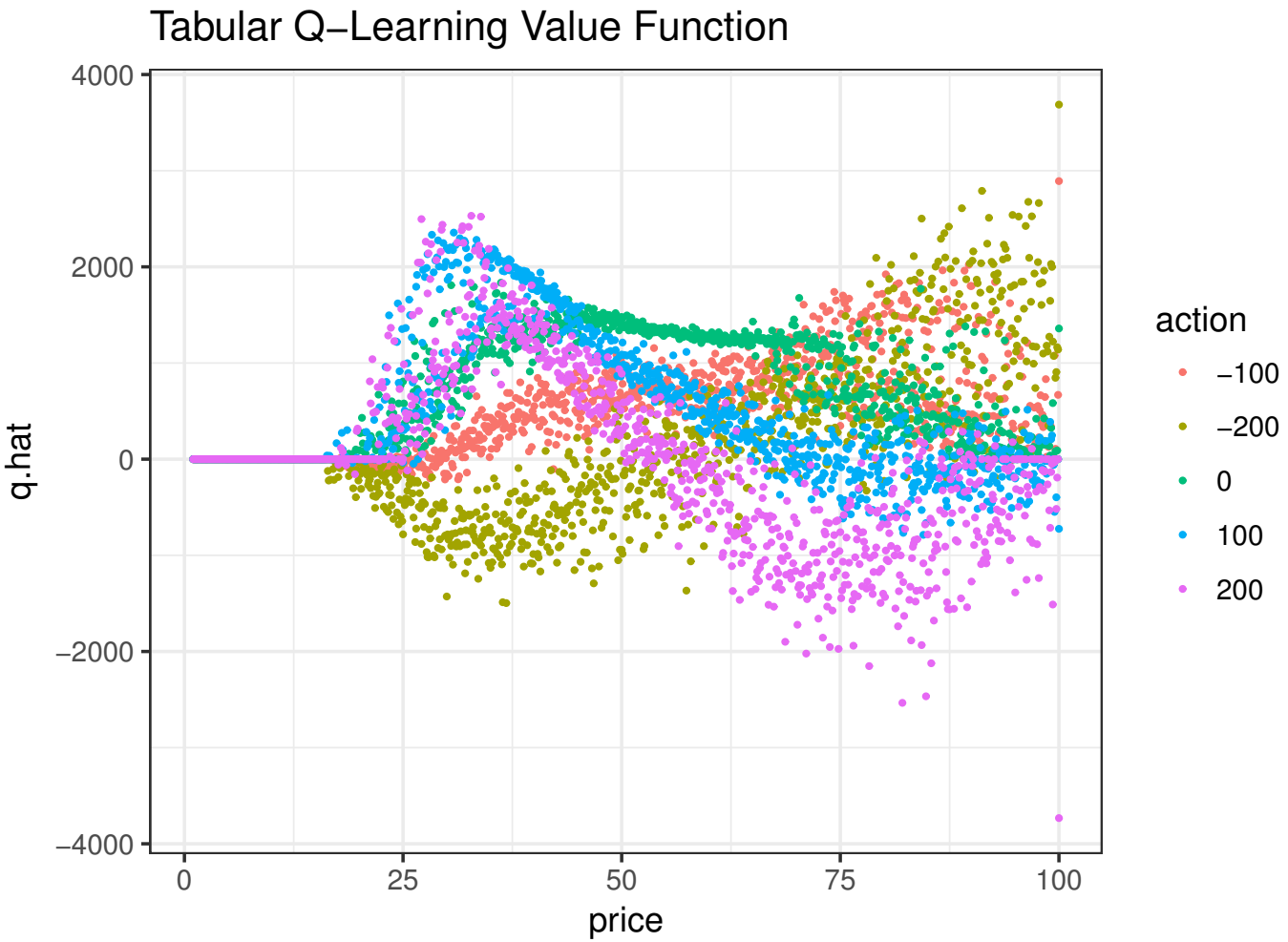


Figure: Value function $p \to \hat{q}((0, p), a)$, where $\hat{q}$ is estimated by the tabular method.

- The tabular method estimates each element $\hat{q}(s, a)$ individually, with no "nearest neighbor" effects or tendency towards continuity.

- In this example, the optimal action choice has a natural monotonicity, which we now describe intuitively.
- Suppose that our current holding is $h = 0$.
- Suppose that for some price $p < p_e$, the optimal action, given $h = 0$, is to buy 100 shares;
- Then for any price $p' < p$, the optimal action must be to buy at least 100 shares.

- For large price values, the tabular value function seems to oscillate between several possible decisions, contradicting the monotonicity property.
- This is simply an aspect of estimation error and the fact that the tabular method hasn't fully converged even after millions of iterations.
- The tabular value function also collapses to a trivial function in the left tail region, presumably due to those states not being visited very often.

- All of these problems are related to the same underlying problem: we have used a finite state space.
- Ultimately, methods which assume a finite state space are doomed to break down. What if we had 10 variables we wanted to use in the state vector, each with 100 possible values? There would be billions of parameters to estimate.
- We strongly advocate for the use of *continuous state spaces* in reinforcement learning.

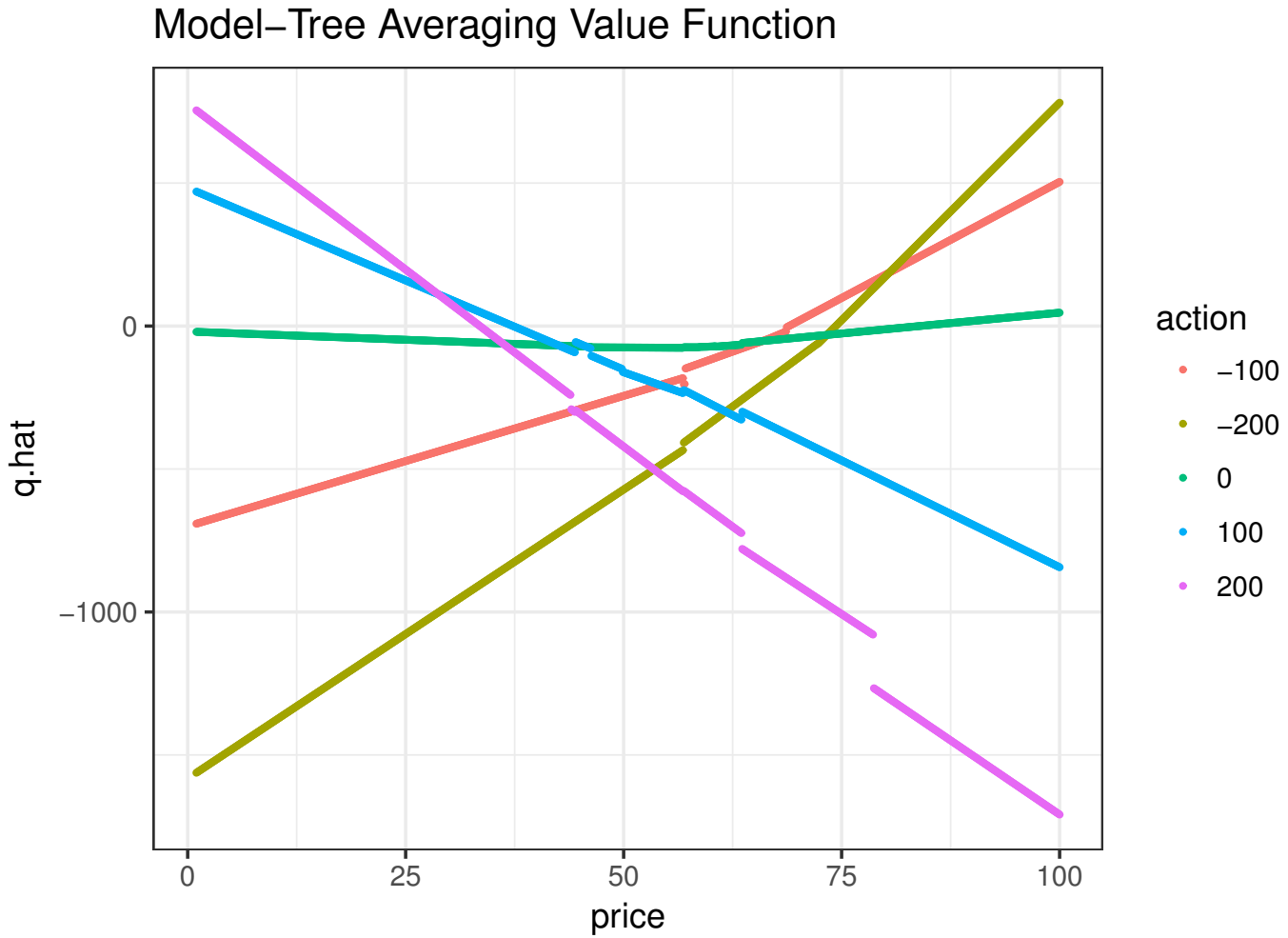


Figure: Value function $p \to \hat{q}((0, p), a)$ for various actions a, where $\hat{q}$ is estimated by using continuous state-space methods. For details, see the preprint on my Courant website.

- The relevant decision at each price level (assuming zero initial position) is the maximum of the various piecewise-linear functions shown in the figure.
- There is a no-trade region in the center, where the green line is the maximum.
- There are then small regions on either side of the no-trade zone where a trade $n = \pm 100$ is optimal, while the maximum trade of ± 200 is being chosen for all points sufficiently far from equilibrium.

- Our testing indicates that the continuous-state-space method not only produces a value-function estimate that is piecewise-continuous, but also outperforms the tabular method in the performance metric.
- Running each policy out of sample for 500,000 steps, we estimate Sharpe ratio of 2.78 for Tabular QL, and 3.03 for continuous-state-space.
- The latter is better able to generalize to conditions unlike those it has already seen, as evidenced by the left-tails in Figs. 2.2–2.3.

Optimal Hedging for Derivatives

- Let's explore applying the above to another problem of interest to traders: hedging an option position.
  We look at the simplest possible example: A European call option with strike price K and expiry T on a non-dividend-paying stock.
- We take the strike and maturity as fixed, exogenously-given constants. For simplicity, we assume the risk-free rate is zero.
- The agent we train will learn to hedge *this specific option* with this strike and maturity. It is not being trained to hedge any option with any possible strike/maturity.

- In any successful application of RL, the state must contain all of the information that is *relevant* for making the optimal decision.
- Information that is not relevant to the task at hand, or which can be derived directly from other variables of the state, does not need to be included.

- For European options, the state must minimally contain the current price S_t of the underlying, and the time

$$\tau := T - t > 0$$

still remaining to expiry, as well as our current position of n shares.

- The state is thus naturally an element of

$$\mathcal{S} := \mathbb{R}_+^2 \times \mathbb{Z} = \{(S, \tau, n) \mid S > 0, \tau > 0, n \in \mathbb{Z}\}.$$

- The state *does not* need to contain the option Greeks, because they are (nonlinear) functions of the variables the agent has access to via the state.
- We expect agents to learn such nonlinear functions on their own as needed.
- This has the advantage of not requiring any special, model-specific calculations that may not extend beyond BSM models.

- We first consider a "frictionless" world without trading costs and answer the question of whether it is possible for a machine to learn what we teach students in their first semester of business school: Formation of the dynamic replicating portfolio strategy.
- Unlike our students, the machine can only learn by observing and interacting with simulations.

- The RL agent is at a disadvantage, initially. Recall that it does not know *any* of the following pertinent pieces of information:
  1. the strike price K,
  2. the fact that the stock price process is a GBM,
  3. the volatilty of the price process,
  4. the BSM formula,
  5. the payoff function $(S - K)_+$ at maturity,
  6. any of the Greeks.

  It must infer the relevant information from these variables, insofar as it affects the value function, by interacting with a simulated environment.
- The results are depicted in Figure 3.1.

Figure: Out-of-sample simulation of a trained agent. We depict cumulative stock, option, and total P&L; RL agent's position in shares (stock.pos.shares), and $-100 \cdot \Delta$ (delta.hedge.shares). Observe that (a) cumulative stock and options P&L roughly cancel one another to give the (relatively low variance) total P&L, and (b) the RL agent's position tracks the delta position even though they were not provided with it.

A second example for Figure 3.1.

- A key strength of the RL approach is that it does not make any assumptions about the form of the cost function (2.4).
- It will learn to optimize expected utility, under whatever cost function you provide.
- As we need a baseline, we define π_{DH} to be the policy which always trades to hedge delta to zero according to the Black–Scholes model, rounded to the nearest integer number of shares.

$$\pi_{DH}(s_t) = \pi_{DH}(p_t, \tau, n_t) := -100 \cdot \text{round}(\Delta(p_t, \tau)) - n_t \quad (3.1)$$

- Previously we had taken multiplier $= 0$ in the function cost(n) representing no frictions.
- We now take multipler $= 5$, representing a high level of friction.
- Our intuition is that in high-trading-cost environments (which would always be the case if the position being hedged were a very large position relative to the typical volume in the market), then the simple policy π_{DH} trades too much.
- One could perhaps save a great deal of cost in exchange for a slight increase in variance.

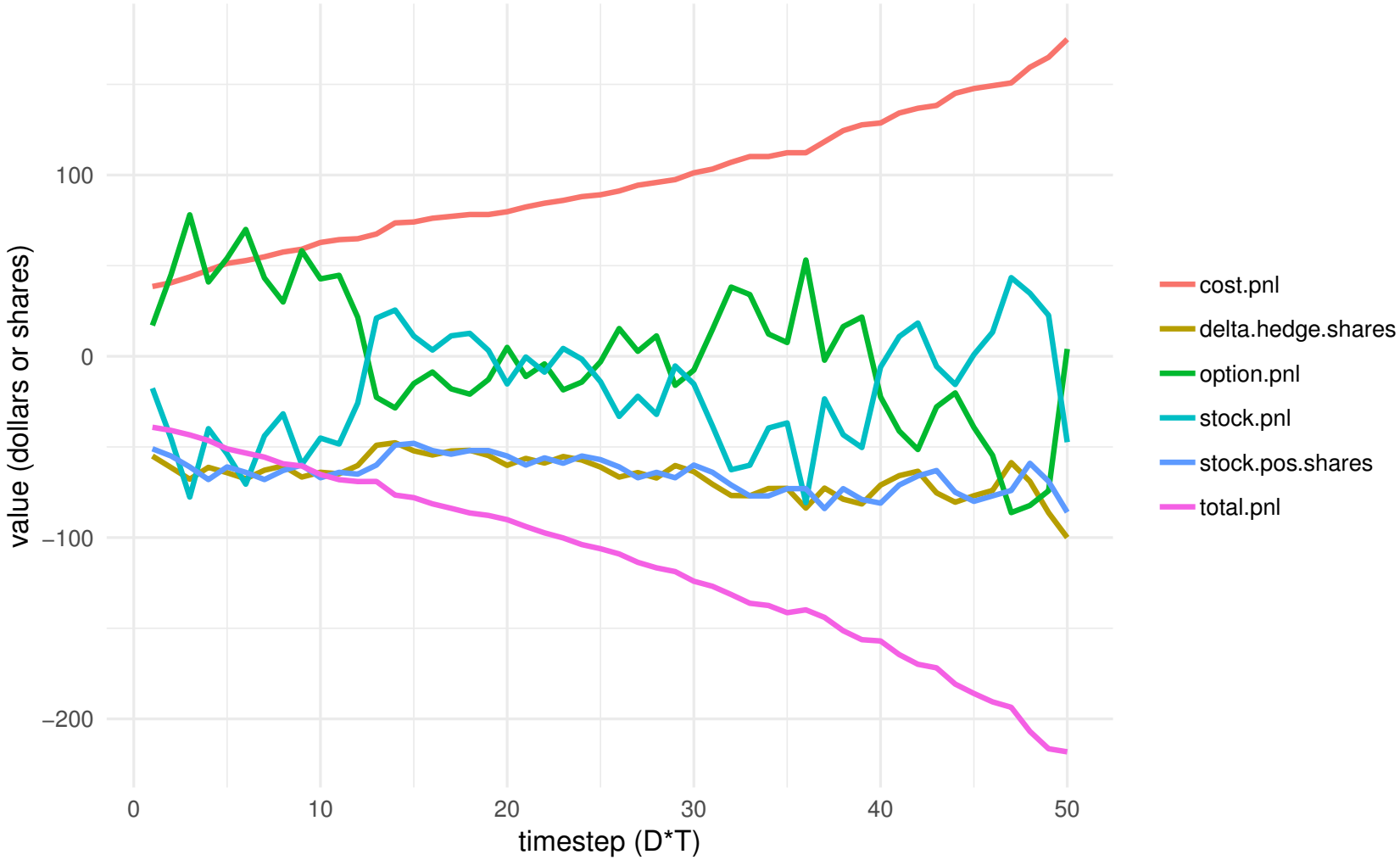
- Given that we are using a reward signal that converges to the mean-variance form of the utility function,

$$\mathbb{E}[w_T] - \frac{1}{2}\kappa\,\mathbb{V}[w_T],$$

  we naturally expect RL to learn the trade-off between variance and cost.
- In other words, we expect it to realize lower cost than π_{DH}, possibly coming at the expense of higher variance, when averaged across a sufficiently large number of out-of-sample simulations (i.e. simulations that were not used during the training phase in any way).
- After training we ran $N = 10,000$ out of sample simulations. Using the out-of-sample simulations we ran a horse race between the baseline agent who just uses delta-hedging and ignores cost, and the RL trained agent who trades cost for realized volatility.

- Figure 3.2 shows one representative out-of-sample path of the baseline agent. We see that the baseline agent is over-trading and paying too much cost.
- Figure 3.3 shows the RL agent – we see that, while maintaining a hedge, the agent is trading in a cost-conscious way.
- The curves in Figure 3.2, representing the agent's position (stock.pos.shares), are much smoother than the value of $-100 \cdot \Delta$ (called delta.hedge.shares in Figure 3.2), which naturally fluctuates along with the GBM process.

Figure: Out-of-sample simulation of a baseline agent who uses policy "delta" or π_{DH}, defined in (3.1). We show *cumulative* stock P&L and option P&L, which roughly cancel one another to give the (relatively low variance) total P&L. We show the position, in shares, of the agent (stock.pos.shares). The agent trades so that the position in the next period will be the quantity $-100 \cdot \Delta$ rounded to shares.



Another example of the baseline agent: policy "delta" or π_{DH}.

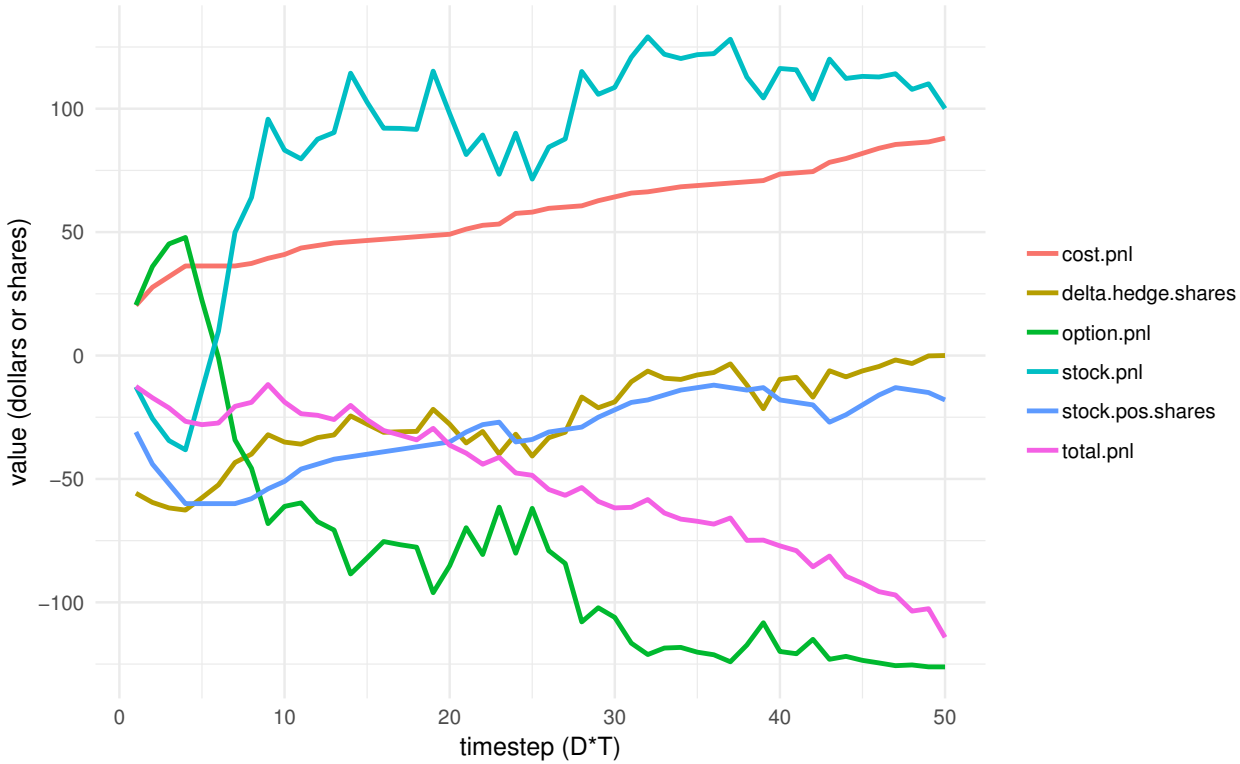


Figure: Out-of-sample simulation of our trained RL agent. The curve representing the agent's position (stock.pos.shares), controls trading costs and is hence much smoother than the value of $-100 \cdot \Delta$ (called delta.hedge.shares), which naturally fluctuates along with the GBM process.

Another example of the trained RL agent.

- Above we could only show a few representative runs taken from an out-of-sample set of $N = 10,000$ paths. To summarize all paths, we compute

$$\begin{aligned} &\text{for } i = 1, 2, \ldots, N: \\ &\quad \text{cost}_i = \text{sum}(\text{total.cost}_{i,t} \ : \ t = 1 \ldots, T) \\ &\quad \text{vol}_i \ = \text{sdev}(\text{total.pnl}_{i,t} \ : \ t = 1 \ldots, T) \end{aligned}$$

  We then plot kernel density estimates (basically, smoothed histograms) of cost_i and of vol_i, each a vector of length N.
- The difference in average cost is highly statistically significant, with a t-statistic of -143.22. The difference in vols, on the other hand, was not statistically significant at the 99% level.

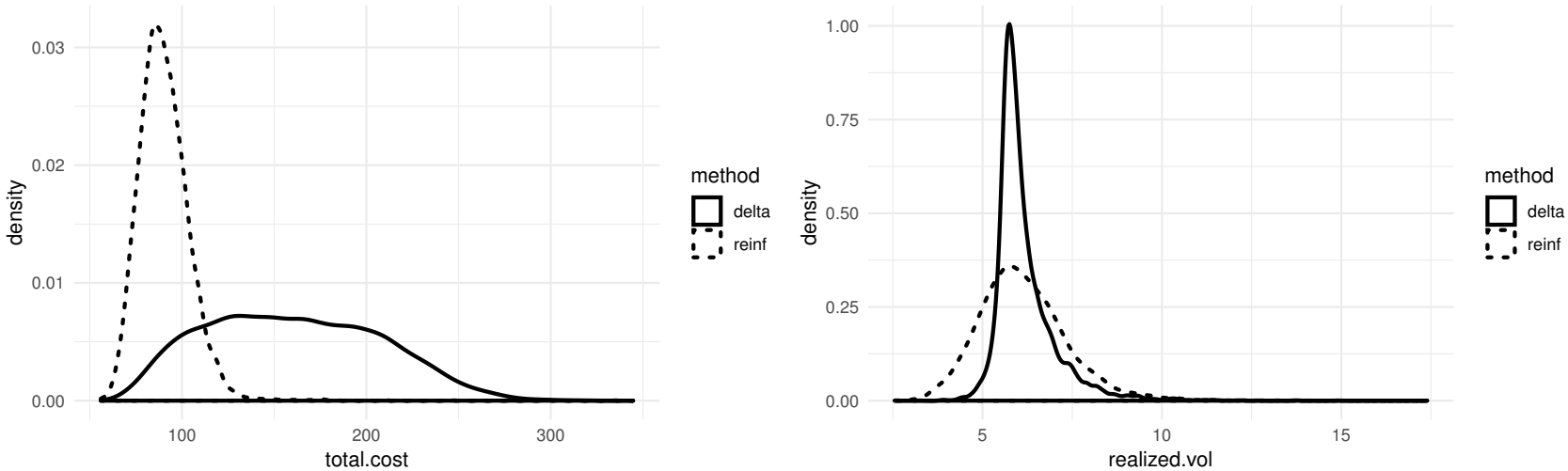


Figure: Kernel density estimates for total cost (left panel) and volatility of total P&L (right panel) from $N = 10,000$ out-of-sample simulations. Policy "delta" is π_{DH}, while policy "reinf" is the greedy policy of an action-value function trained by RL. The "reinf" policy achieves much lower cost (t-statistic $= -143.22$) with no significant difference in volatility of total P&L.

- To summarize, it does seem that machines can find arbitrage opportunities in data, when they exist, and can become long-term greedy in the presence of costs.
- The subject is in its infancy, and we are a long way from the "Skynet" of trading. We share a few things we have learned by trial-and-error :^).
- It is generally most useful to work in continuous state space.
- Most (optimal) value functions in finance are continuous functions of the state, and some have monotonicity properties dictated by economics. Some are even smooth or piecewise-smooth.
- Paying careful attention to the actual nonlinear-function-approximation (aka supervised learning) that is used to approximate the value function, and using a prior that encourages the desired properties, is quite useful.

- The same methods can be used effectively to hedge derivatives in illiquid markets where trading costs are an extremely important contribution to wealth.
- Reinforcement learning agents can learn to price and hedge derivatives in markets where perfect replication would be impossible, or when perfect replication would be too costly.
- It can do this with just a good simulator, no other "help" from humans.


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