Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-023-04657-y

ORIGINAL RESEARCH

Structured products dynamic hedging based on reinforcement learning

Hao Xu[1] · Cheng Xu[2] · He Yan[3] · Yanqi Sun[4]

Received: 30 September 2022 / Accepted: 21 June 2023

Abstract
In the Black–Scholes model proposed in 1973, an investor can use a continuously rebalanced dynamic strategy to hedge the risk of a certain option, assuming that the underlying asset's price is subject to geometric Brownian motion (a continuous-time stochastic process where the logarithm of the variable follows a Brownian motion) and the market is complete and frictionless, which is unrealistic due to the continuous changes in asset prices. The application of reinforcement learning (RL) in finance includes a variety of decision-making problems such as hedging, optimal execution, and portfolio optimization. RL can make full use of historical data or generate more data than other theories used to make decisions in finance such as stochastic control theory. There will be fewer assumptions and better performance with exploration and exploitation. In this article, we propose a reinforcement learning-based model that can help investors dynamically hedge financial products in discrete time using complex structured products: the Phoenix option (a note that only pays a coupon if the price of the underlying asset is above a certain barrier and redeems if the price breaches an autocall barrier) as an example in this paper. This model is highly expandable and can set an objective function according to the investor's preferences; for example, the Sharpe ratio (a measure of risk-adjusted return that compares the return of an investment with its risk) is very lightweight because we do not assume the existence of an optimal hedging strategy.

Keywords Structure products · Dynamic hedging · Reinforcement learning

1 Introduction

In the Black–Scholes model proposed in 1973, an investor can use a continuously rebalanced dynamic strategy to hedge the risk of a certain option. The aim of this dynamic hedging method is to construct a risk-neutral portfolio with respect to a certain parameter in this model, that is, the Greeks. However, in practice, these parameters change continuously because the price of the underlying asset changes continuously. This continuous hedging strategy is unrealistic in the case of transaction costs. Therefore, the adjustment time for the position is discrete. Research on this hedging strategy depends on the expansion of the Black–Scholes model and relaxation constraints (Taleb 1997; Caldentey and Haugh 2006; Fliess and Join 2010).

Machine learning (ML) is part of artificial intelligence, and is designed to make predictions or decisions. Recently, ML has been increasingly applied to finance. Although many studies have used ML methods, the specific algorithms and purposes vary greatly. Some models use ML to solve the mathematical problems in the Black–Scholes

Hao Xu and Cheng Xu contribute to the manuscript equally.

✉ Yanqi Sun
bbmaksun@gmail.com

Hao Xu
kenji.xu@outlook.com

Cheng Xu
Cheng.Xu@xjtlu.edu.cn

He Yan
hyan1@babson.edu

1 Carson College of Business, Washington State University, Pullman, USA

2 Department of Strategic Management and Organisations, International Business School Suzhou, Xi'an Jiaotong-Liverpool University, 8 Chongwen Road, Suzhou Industrial Park, Suzhou, Jiangsu, China

3 Babson College, Wellesley, MA, USA

4 School of Economics and Management, Beijing Institute of Petrochemical Technology, 19 Qingyuan North Road, Daxing District, Beijing, China

model, while others break the Black–Scholes model framework and directly use ML to establish a new framework to analyze the hedge strategy.

In the Black–Scholes model, it is important to solve partial differential equations (PDEs). Several methods have been proposed to solve these problems. Barucci et al. (1996) used the Glaerkin method and ANNs to solve the Black–Scholes PDE. Chan-Wai-Nam, Mikael, and Warin used ML to solve semilinear PDEs. Joo and Moon (2021) proposed a quantum variational PDE solver with the aid of machine-learning schemes to synergize two emerging technologies in mathematically hard problems.

Some models are not completely dependent on the Black–Scholes model. Gammerman and Vovk(2007) used ML algorithms such as support vector machines (SVM), kernel ridge regression (KRR), and kernel nearest neighbors (KNN) to hedge the predictions. Das and Padhy (2017) proposed a hybrid model that combines the Black–Scholes option pricing model, Monte Carlo option pricing model, and the finite difference method with support vector regression (SVR) and extreme learning machine-based regression models. Shin and Ryu (2012) presented a methodology for a dynamic option-hedging strategy using an ANN to enhance hedging performance.

Recently, the development of reinforcement learning (RL) has introduced new methods for pricing and hedging. The first person to use reinforcement learning for hedging was Halperin (2020), who minimized the terminal variance of the hedge portfolio based on the Markowitz portfolio theory to obtain the optimal solution. Kolm and Ritter (2019) built a system that can automatically learn how to optimally hedge an option in a fully realistic setting using RL. Buehler et al. (2019a, b) used deep reinforcement machine-learning methods to establish frameworks for portfolio hedging.

Research on hedging strategies is accompanied by a relaxation of constraints to be closer to the real world. The power of RL enables us to train an agent to automatically obtain an optimal hedging strategy, even in a complex environment. In this article, we propose a simple reinforcement learning method that can train an agent and obtain a hedging strategy that can facilitate them getting more rewards than the rewards from the "buy and hold" policy. Rewards can be customized according to varied preferences, like the structured product itself. Specifically, the reward function of agents trained in this study is the Sharpe ratio, which is one of the most used investment return indices in finance. The definition is the difference between the expected return on the asset and risk-free return divided by the standard deviation of the asset return.

Compared to traditional methods, our approach is closer to market reality as an explicit pricing formula, and the existence of a perfect dynamic hedging strategy is unnecessary. Hedging is essentially an interaction with the environment, and the interaction involves acquiring new knowledge and using the knowledge to improve performance. The trade-off between the two is one of the most fundamental trade-offs in nature, and optimal performance usually requires a balance between exploratory and exploitative behavior (Berger-Tal et al. 2014). The key to reinforcement learning algorithms is to obtain this balance, whereas other machine learning algorithms, such as SVM and LSTM, do not focus much on this issue (Andrew 1999). In addition, previous studies of reinforcement learning hedging strategies have aimed to study a variety of different algorithms, such as Q-learning (Halperin 2020), SARSA (Kolm and Ritter 2019), deep Q-learning network, and proximal policy optimization (Du et al. 2020); their objective function is often a simple yield to maturity, whereas in practice, investors are not only concerned with yield, but risk, that is, variance, which should also be considered, and our model does this.

This study contributes to the literature in several ways. First, the proposed method is quite simple as an explicit pricing formula, and the existence of a perfect dynamic hedging strategy is unnecessary. Second, our objective is closer to that of the real investment world. We considered the psychological changes of investors in the investment process; that is, we controlled for the Sharpe ratio of each period. We also considered cases in which investors can focus only on the Sharpe ratio at maturity or give it more weight. Furthermore, other performance measures, such as the Sortino ratio, which is a risk-adjustment metric used to determine the additional return for each unit of downside risk (Sharpe 1966), instead of the Sharpe ratio, are also applicable in this model.

The structure of the paper is as follows:

In Sect. 2, we review the literature on asset hedging strategies.

In Sect. 3, we introduce structured products and Phoenix options and their features in detail.

In Sect. 4, we introduce the basics of reinforcement.

In Sect. 5, we train the agent via a simulation and prove that the hedging strategy based on RL is better.

In Sect. 6, we make some extensions and discussions about our method.

2 Literature review

In this section, we highlight the main research streams on hedging strategies in the literature.

In the Black–Scholes model, it is important to solve partial differential equations (PDEs). Several methods have been proposed to solve these problems. Barucci et al. (1996) used the Glaerkin method and ANNs to solve the Black–Scholes PDE. Chan-Wai-Nam, Mikael, and Warin used ML to solve semilinear PDEs. Joo and Moon (2021) proposed a quantum variational PDE solver with the aid of machine-learning schemes to synergize two emerging technologies in mathematically difficult problems. Based on the Black Scholes model, researchers have extended this model by relaxing some unrealistic assumptions to explain the implied volatility smile as well as the skewness and kurtosis in the return distribution. The first is the model of the jump-diffusion process proposed by Merton (1976). The second is the stochastic volatility model (Hull and White 1987). The last model is the local volatility model, developed by Dupire (1994). Although these models are closer to reality than the Black–Scholes model, they still have limitations. In addition to being difficult to compute for complex financial products, they are not very effective for short-term pricing (Bakshi et al. 1997).

Another study on hedging strategies mainly used machine-learning methods. Gammerman and Vovk (2007) used ML algorithms such as support vector machines (SVM), kernel ridge regression (KRR), and kernel nearest neighbors (KNN) to hedge the predictions. Das and Padhy (2017) proposed a hybrid model that combines the Black–Scholes option pricing model, Monte Carlo option pricing model, and the finite difference method with support vector regression (SVR) and extreme learning machine-based regression models. Shin and Ryu (2012) presented a methodology for a dynamic option-hedging strategy using ANN to enhance hedging performance. Reinforcement learning has unique significance in the study of hedging strategies because of its unique characteristics. The first person to use reinforcement learning for hedging was Halperin (2020), who minimized the terminal variance of the hedge portfolio based on Markowitz portfolio theory to obtain the optimal solution. Kolm and Ritter (2019) built a system that can automatically learn how to optimally hedge an option in a fully realistic setting using RL. Other studies that used reinforcement learning to study hedging strategies include Buehler et al. (2019a, b), Du et al. (2020), and Gu (2022). They have made many extensions to the model to make it more optimal, but their reward function is not free from the Black–Scholes model and does not consider the realistic reward function that investors need.

The proposed model in this study is closer to the real investment world. We also considered the psychological changes of investors in the investment process; that is, we controlled for the Sharpe ratio of each period. We also consider cases in which investors can focus only on the Sharpe ratio at maturity or give it more weight.

3 Structured products

Structured products can be understood as financial products tailored to clients. Officially, structured products are financial instruments whose performance or value is related to that of the underlying asset, product, or index. These are pre-packaged and non-traditional products that are used as alternatives to direct investment. A structured product can be seen as a product package using three main components: bonds, one or more underlying assets, and derivative strategy. Thus, structured products are essentially combinations of multiple financial derivatives.

Some simple structured products include market indices, and most structured products are more complex, such as the Phoenix option used in this study.

Structured products are now well established in most countries throughout Europe and are provided by major banks, insurance groups, and various other organizations.

Based on the BSM model, publishers can offer various structured products without taking market risks. We used a relatively complex structured product in this study as an example to illustrate the power of reinforcement learning. Specifically, we study the dynamic hedging strategy for a structured product called the Phoenix option, which is an autocallable note, whose features are described below.

3.1 Phoenix option features

First, we introduce the important time points for the Phoenix options. Phoenix options set interesting observation dates, usually once a month, and the closing price of this day determines your contingent interest payment. In addition to the observation date of interest, there is an auto call observation date, usually once a season, the closing price of which day will determine whether the product would be automatically called; maturity is uncertain because Phoenix options may be automatically called.

We then present the payments regarding the above important time points: contingent interest payment, automatically callable, and contingent repayment of the principal amount at maturity.

(1) Contingent interest payment.

In every interest observation date, if the closing price of the underlying equity is equal to or greater than a certain price, called the interest barrier, the investor will receive a contingent interest payment, which is a certain proportion of the principal in the auto-call observation dates.

(2) Automatically callable.

On every auto call observation date, the notes will be automatically called if the closing price of the underlying equity is less than the interest barrier or greater than a certain price called the trigger price. When the notes are automatically called, the investor receives a contingent repayment like the contingent repayment of the principal amount at maturity in (3). In most cases, the trigger price is equal to the interest barrier.

(3) Contingent repayment of principal amount at maturity.

The investor obtains the principal back if the notes have not been called and the final price is greater than the trigger price by maturity. Otherwise, if the final price is less than the trigger price, the investor will take the loss himself; that is, he will only get the cash equivalent whose value is less than the principal in this case.

3.2 Examples

This section we give an example to better understand Phoenix option.

For convenience, we do not consider the risk-free interest rate.

First, we summarize some important parameters used in the examples (see Table 1).

(1) The closing price of the underlying equity is equal to or greater than the initial price on the first auto call observation date.

Date	Closing price	Payment per note
1st interest observation date	52	15
2nd interest observation date	35	0
3rd interest observation date and 1st autocall observation date	51	1015
Total payment: 1030		

(2) Notes are not called and the final price is greater than the trigger price.

Date	Closing price	Payment per note
1st interest observation date	41	15
2nd interest observation date	30	0
3rd interest observation date and 1st autocall observation date	42	15
4th to 11th interest observation date	(40, 50)	120
Valuation date (the final date)	45	1015
Total payment: 1165		

(3) Notes are not called and the final price is less than the trigger price.

Date	Closing price	Payment per note
1st interest observation date	41	15
2nd interest observation date	30	0
3rd interest observation date and 1st autocall observation date	42	15
4th to 11th interest observation date	(40, 50)	120
Valuation date (the final date)	30	30×20 shares $= 600$
Total payment: 750		

Table 1 Parameters of Phoenix options

Principal amount	1000 dollars
Term	12 months
Initial price	50 dollars
Contingent interest payment	15 dollars
Interest observation dates	Monthly
Auto call observation dates	Seasonally
Interest barrier	40 dollars (80%)
Trigger price	40 dollars (80%)
Averaging dates	N/A

4 Reinforcement learning

Reinforcement learning (RL) is a type of machine learning algorithm which can be seen as a third machine learning paradigm along with side-supervised learning and unsupervised learning. In other words, RL learns the mapping from state to action. For example, given the current position, underlying asset price, and some necessary known information, this mapping can be used to take optimal action, that is, the hedging strategy. Mathematically, RL is formalized using ideas from dynamical systems theory, specifically as the optimal control of an incompletely known Markov decision process (MDP).

The main concepts of RL can be summarized as exploration and exploitation. Exploration means that the RL method tries different actions which have not been tried before by the RL method to obtain better actions, whereas exploitation means that the RL method prefers actions that can offer more expected rewards. In RL, researchers often use the epsilon-greedy algorithm to balance exploration and exploitation, which we introduce in the next section.

Recently, an increasing number of studies have begun to focus on the application of reinforcement learning in option hedging (Du et al. 2020; Kolm and Ritter 2019; Cao et al. 2021; Vittori et al. 2020), optimal execution (Nevmyvaka et al. 2006), portfolio optimization (Almahdi and

Yang 2017; Yu et al. 2019), and market making (Spooner et al. 2018; Lim and Gorse 2018; Ganesh et al. 2019) etc.

Some important elements in RL are shown as follows:

(1) Random variables

A random variable is a variable whose value is unknown or is a function that assigns values to each of the experimental outcomes. In RL, St and At are random variables.

(2) Stochastic process

A stochastic process is a collection of random variables indexed by a mathematical set.

(3) Markov chain/process

A Markov chain process is a stochastic process with Markov properties. In discrete-time cases, the Markov property can be formulated as follows:

$$P(X_n = x_n | X_{n-1} = x_{n-1}, \dots X_0 = x_0) = P(X_n = x_n | X_{n-1} = x_{n-1}),$$

where {Xn} is a stochastic process.

The intuition for this definition is that the evolution of the Markov process in the future depends only on the present state and not on history. Thus, the Markov process is also called a process with memoryless properties.

(4) Markov decision process

A Markov decision process (MDP) is a process with reward and action functions. In RL, we train an agent by simulating agent-environment interactions. The agent is the learner or the decision maker, who selects actions at every step, and the environment responds to these actions with rewards and presets new states to the agent. These interactions are shown in Fig. 1.

RL's job is to obtain policy π, which is, in essence, a mapping from states to actions. For example, maintaining the hedging position at a constant 0 is a policy. The optimal policy π* satisfies the assumption that the agent can maximize the expectation of Gt based on this policy.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{i=1}^{\infty} \gamma^{i-1} R_{t+i},$$

where γ is the discount factor, γ ∈ [0,1], and Rt is the reward after every step. Note that the limit that γ < 1 is not a necessity in recent studies, which we discuss later.

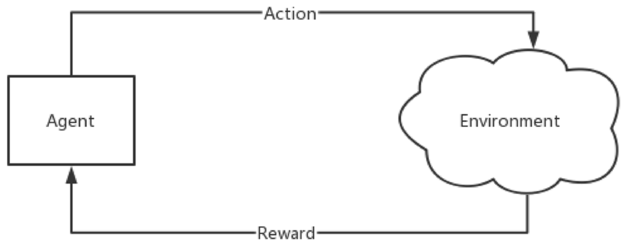

Fig. 1 The agent-environment interaction

There are two important value functions in RL: state-value and action-value.

The state-value and action-value functions are defined as follows:

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

where S_t represents the state at time t and A_t represents the action at time t according to policy π.

We find that v is the weighted average of q; that is,

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$$

and

$$q_\pi(s,a) = \sum_{r,s'} p(s', r|s,a)(r + \gamma v_\pi(s'))$$

where the transition probability is p(s′, r| s, a), which means that in state s, the agent takes action a, the probability of obtaining a reward r, and the state transition to s′ is p(s′, r| s, a).

Using these two equations, we can obtain two important recursion formulas known as the Bellman equation.

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s,a)[r + \gamma v_\pi(s')]$$

$$q_\pi(s,a) = \sum_{s',r} p(s', r|s,a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a') \right].$$

We can then obtain the optimal state-value and action-value functions as follows:

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

$$q^*(s,a) = q_{\pi^*}(s,a) = \sum_{s',r} p(s', r|s,a)\left[r + \gamma \max_{a'} q^*(s', a')\right],$$

where v^* and q^* represent the optimal state-value and action-value functions, respectively, and π* is the optimal policy.

RL has different algorithms, such as Q-learning, SARSA, deep Q Network (DQN). Q-learning aims to maximize the Q function in the Bellman equation, and DQN is an improved version of Q-learning, mainly to enhance generality by introducing neural networks. This study used the SARSA algorithm, which we recalled.

SARSA is temporal-difference (TD) learning, which is a combination of Monte Carlo (MC) and dynamic programming (DP) ideas. First, let us recall DP and MC.

In RL, DP algorithms are obtained by turning Bellman equations into assignments, that is, updating rules by improving approximations of the desired value functions. Briefly, we can construct a fixed-point equation as follows:

$$v_{k+1} = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)(r + \gamma v_k),$$

where Vk is a column vector (Vk(S1), Vk(S2)….Vk(Sn))T

MC methods are ways of solving RL problems based on average sample returns. They approximated Gt by sampling, as follows:

$$v_\pi(s) = E_\pi[G_t|S_t = s]$$
$$v(S_t) \leftarrow v(S_t) + \alpha(G_t - v(S_t)).$$

TD methods combine the benefits of DP (bootstrapping) and MC (sampling); that is, TD methods can learn directly from raw experience and update the policy after each step without waiting until the end of each episode. The policy updates of TD methods can be summarized as follows:

$$v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(s_{t+1})|s_t = s]$$
$$v(s_t) \leftarrow v(s_t) + \alpha[R_{t+1} + \gamma v(s_{t+1}) - v(s_t)].$$

SARSA and Q-learning are the two main TD algorithms. The policy update is as follows.

$$SARSA:$$
$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \alpha[r_{t+1} + \gamma q(s_{t+1}, a_{t+1}) - q(s_t, a_t)]$$
$$Q - Learning:$$
$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a q(s_{t+1}, a) - q(s_t, a_t)]$$

The update is performed after every transition from a nonterminal state. SARSA uses a quintet tuple: (St,At,Rt + 1,St + 1,At + 1). This was the origin of the name.

In this study, we use the SARSA algorithm to train the agent, as SARSA learns the Q-value based on the action performed by the current policy instead of the greedy policy, which means that the SARSA algorithm can find the optimal hedging strategy faster. Although the characteristics of SARSA would make it impossible to find a truly optimal solution, we chose SARSA over Q-learning because the structured products in our portfolio are quite complex and no optimal hedging strategy necessarily exists in our assumptions. The other methods are discussed in Sect. 6.3.

5 Training via simulation

In this section, we introduce our method based on an environmental simulation. Before introducing important settings, we provide an overall training framework. First, we assume that the agent holds one share option and can but never trade it, which means that the position of option in the portfolio is always constant at 1. At every interest observation date and at time 0, the agent will decide to buy or sell a certain number of underlying equity depending on the current state and Q-table. The Q-table can be understood figuratively as an agent's action guide, which is essentially a discrete function; that is, given a state (S_t, τ, pos), we can search the Q-table for an optimal action.

We performed this training by simulating episodes. In each simulation, the price of the underlying asset was generated randomly. Some of the necessary parameters are predetermined, whereas the paths are not. The details are presented in Sect. 2.1.

In the training, the current state and Q-Table are all the agents know, while in practice, they may know more than that, a certain structured product's rules, for example. While we can add more information in the state space to make the agent aware of this, it is unnecessary because an increase in the number of variables in the state space will lead to an extreme increase in the number of episodes; that is, we may need more episodes to converge. Therefore, we do not bother to do so, and only some confidential variables in the state space are sufficient.

Moreover short selling is allowed in our case, and investors can lend or borrow at a risk-free interest rate.

5.1 Set up

First, we assume that the underlying equity obeys the geometric Brownian motion (GBM); that is, the price of the underlying equity is a stochastic process S_t, satisfying the following stochastic differential equation (SDE):

$$dS_t = rS_t dt + \sigma S_t dW_t,$$

where W_t is Brownian motion, and r and σ are constants that represent the risk-free interest rate and volatility, respectively.

Solving the SDE, we can obtain the analytic solution:

$$\ln S_t - \ln S_0 = \left(r - \frac{1}{2}\sigma^2\right)t + \sigma W_t,$$

where S_0 is the initial price of the underlying equity.

Table 2 Some important parameters of Phoenix options in training

Principal amount	100 dollars
Term (T)	12 months
Initial price (S_0)	100 dollars
Contingent interest payment (CPMT)	5 dollars
Interest observation dates	monthly
Auto call observation dates	seasonally
Interest barrier	80 dollars (80%)
Trigger price	80 dollars (80%)
Averaging dates	N/A
Risk-free interest rate (r)	3%
Volatility of underlying equity	15%

Then we set some important parameters of Phoenix options (see Table 2).

It is necessary to point out that the absolute value of these parameters or results are meaningless, unlike the relative values. For example, in the following section, we calculate the Sharpe ratio of the raw portfolio as 1.87. In practice, if an asset's Sharpe ratio is greater than 1, it is worth investing in. However, what matters is the comparison between the Sharpe ratio and 1.87 after adopting the hedging strategy.

Next, we set up the action function A_t, which is our policy. In general, A_t is determined by the policy function. One of the challenges that arise in RL, and not in other types of learning, is the trade-off between exploration and exploitation, as mentioned before. To strike a balance between them and fully utilize them, a greedy policy called ε-greedy policy is defined as follows:

$$\pi_{\varepsilon-greedy}(s) = \begin{cases} random(a) & u < \varepsilon \\ \arg\max\limits_{a} q(s,a) & u \geq \varepsilon \end{cases},$$

where u is a random number generated before choosing which action to take, and random(a) indicates that if u < ε, this policy will choose a random action from the action space. In our training, ε was an iteration-dependent variable instead of a constant. As the training progressed, ε continued to decay. First, we set a relatively high ε of 0.5 for example, and the final ε was 0.01. Assuming the number of episodes is 10k, then at the ith episode, εt is 0.5 − 0.000049 (i − 1). Through this setting, the model will explore the state space thoroughly at the beginning of training, and then with ε becoming increasingly smaller, the model exploits the Q-Table more.

In our method, we used the Sharpe ratio of the portfolio to define the reward. First, we consider the Sharpe ratio. The Sharpe ratio is a measure of risk-adjusted return, named after William F. Sharpe (1966). It is one of the most used investment return indices in finance. It is defined as the difference between the expected return on the asset and the risk-free return divided by the standard deviation of the asset return:

$$Sharpe\_Ratio = \frac{E[R - R_f]}{Stddev[R]},$$

where E[R-R_f] represents the difference between asset returns and the risk-free return and Std[R] is the standard deviation of the asset return.

Subsequently, we trained the agent to hedge at every interest observation date (monthly, in this case). In practice, investors are not only concerned about the return at the time of maturity, but are also relatively sensitive to the return and risk in the holding period. Some indices, such as maximum pullback and maximum-pullback days, may also constitute the desired reward function in other cases. However, in this case, as we can obtain a reward at every step, we can use the Sharpe ratio to control the return and risk of each period. Besides the risk and return at maturity is still pivotal; so, we can use the weight related to the due time to express varying importance. Different settings of weight are also important in the training, and many studies have focused on such parameters, which will be discussed later.

The agent hedges the option on every interest observation date, and we use a function of distance to maturity and the Sharpe ratio of the asset portfolio from t = 0 to the current period to define the reward function, that is,

$$\tau = T - t$$
$$R_{t+1} = sharpe\_ratio(\{X_i\}_{i=0}^{t}) \times \frac{12 - \tau}{12 \times (\tau + 1)}.$$

The state function matters because the size of the state space directly determines the convergence speed, and the state must include all information known to the agents. Therefore, according to Kolm and Ritter (2019), the state of every step is a triplet:

$$(S_t, \tau, pos),$$

where S_t is the underlying equity price at time t; τ is the distance to maturity T-t; and pos is the underlying equity position at time t. In the real world, as previously discussed, the agent knows more than the current state function. However, this triplet is sufficient for the agent to make decisions, owing to the power of reinforcement learning.

Finally, we define the value of portfolio Xt as follows:

$$X_{t+1} = pos_t \times S_{t+1} + OPT_{t+1} + cpmt + e^{\frac{r}{T}}(X_t - pos_t \times S_t - OPT_t)$$
$$pos_{t+1} = pos_t + a_{t+1},$$

where OPTt is the price of the Phoenix option at time t, given the underlying equity price St + 1, which is calculated using Monte Carlo Simulation, and cpmt is the contingent interest payment on every interest observation date. This formula is an intuition from that at time t, the investor takes action a_t, his position becomes pos_t, his cash is (Xt − post*St-OPTt), and at time t + 1, he may get or pay the interest.

5.2 Dynamic hedging strategy

As previously mentioned, in our training, the underlying asset price is totally stochastic, and in practice, it is generated at every step rather than predetermined. Our method is to obtain a strategy called the Q-Table.

First, we calculate the Sharpe ratio at maturity without taking hedging action (see Fig. 2). We simulate 10k times and use the average Sharpe ratio as the benchmark.

We conducted 10k simulations and the average Sharpe ratio is 1.87.

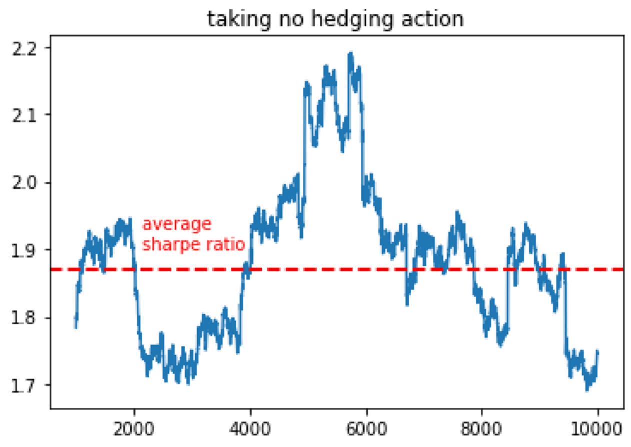

Fig. 2 The Sharpe ratio at maturity without taking any hedging action

We then trained the agent using 100k episodes. In other words, we simulated 100k underlying equity price paths. In fact, this quantity is far from sufficient. Although we maintain one decimal place when we calculate St and pos,, the number of state spaces of St and pos can be over 500 and 2000, respectively, given that St ranges from 75 to 125 and pos ranges from − 100 to 100. Therefore, there can be more than 1 m different states without considering the due time. It is necessary to mention that if one state never appears in the Q-Table before, the hedging strategy is zero, that is, it takes no action. The results are shown in Fig. 3. The result is bound to show a certain randomness, but it is also apparent that there is an upward trend because our simulation is "on-policy" too, which means every episode almost surely generates a totally different path. The performance was below the benchmark until 75k episodes.

6 Extensions and discussion

Our method can also be modified according to different preferences. In this section, we introduce two cases. First, we consider different time weights; that is, some investors may only focus on performance at maturity, so we can assign more weight to the reward at maturity. Second, we consider other performance indices, such as the Sortino ratio, rather than the Sharpe ratio. In practice, investors can define various reward functions to satisfy different needs.

6.1 Time weight

In 2.1, we define reward function as follows:

$$\tau = T - t$$

$$R_{t+1} = sharpe\_ratio(\{X_i\}_{i=0}^{t}) \times \frac{12-\tau}{12\times(\tau+1)}.$$

In practice, investors have different preferences. One may only focus on the Sharpe ratio at maturity, and assign more weight to the reward at maturity. Thus, we can use the following definitions.

$$\tau = T - t$$

$$R_{t+1} = sharpe\_ratio(\{X_i\}_{i=0}^{t}) \times \frac{12-\tau}{12\times(5\tau+1)} \tag{1}$$

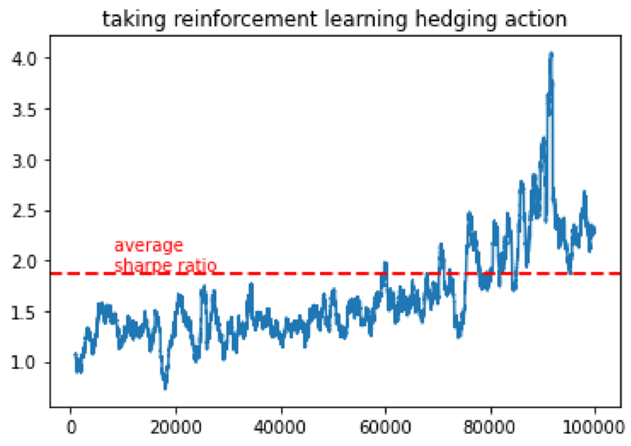

Fig. 3 The Sharpe ratio at maturity taking reinforcement learning hedging action

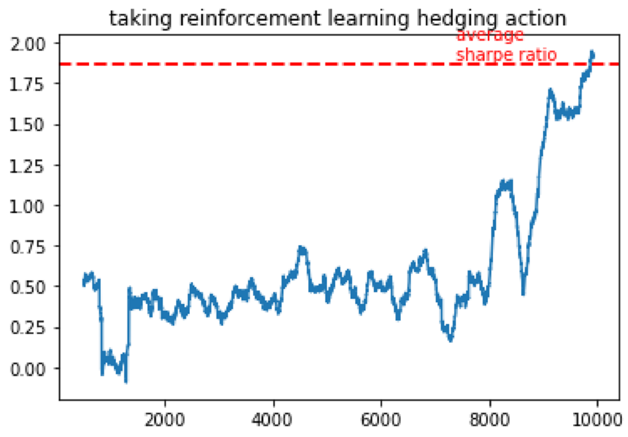

Fig. 4 Performance using reward function in Eq. 1

τ	12	11	10	9	8	7	6	5	4	3	2	1	0
$\frac{12-\tau}{12\times(\tau+1)}$	0	0.007	0.152	0.025	0.037	0.052	0.071	0.097	0.133	0.188	0.278	0.458	1
$\frac{12-\tau}{12\times(5\tau+1)}$	0	0.001	0.003	0.005	0.008	0.011	0.016	0.022	0.032	0.047	0.076	0.153	1

It is clear that by this definition, our method will focus more on the Sharpe ratio at maturity.

The result is shown in Fig. 4.

In this training, less than 10k iterations made the Sharpe ratio at maturity over the benchmark because the weight at maturity is relatively high.

In other methods, we can set all weights equal to 0 except the weight at maturity; we cannot do so in RL because of the essence of RL. However, we can use functions that approach zero to achieve this goal.

In traditional RL, the discount rate γ is typically set between 0 and 1. The discount rate determines the present value of future rewards. If $\gamma = 0$, the agent is myopic. In general, acting to maximize immediate rewards can reduce access to future rewards, so that the return is reduced. We cannot guarantee that we can obtain the same reward the next time we take the same action because the environment is stochastic. The more we consider the future, the greater the uncertainty. Therefore, in most cases, γ was set between 0 and 1 to avoid considering the future. In addition, the range of γ is partly related to mathematical rationality. If a problem is continuous, γ greater than or equal to 1 may render the result unable to converge.

Some studies have begun to use γ values greater than 1. Pitis (2019) provided normative justification for a departure from the traditional MDP setting that allows for a state-action dependent discount factor that can be greater than 1. Bao et al. (2008) use an incremental sequence as the discount factor to tackle the limitations of normal policy-gradient estimation methods on the imbalance of bias and variance.

In our method, we use an incremental state-dependent discount factor to place more weight on the reward at maturity.

6.2 Performance measures

Flexible performance measures allow for the training of agents to better match investor preferences. Specifically, different indices can be used for various purposes. For example, we can use the Sortino ratio, which is a variation of the Sharpe ratio, and differentiate harmful volatility from total overall volatility by using the asset's standard deviation of negative portfolio returns.

In addition, different utility functions are used. For example, the utility function is used in investment.

$$U = E(r_P) - \frac{1}{2}A\sigma_P^2,$$

where $E(r_P)$ is the expected return of the portfolio, and σP is the standard deviation of the portfolio's excess return. A is an index of investor risk aversion. The extent to which variance lowers utility depends on A. Risk aversion has a major impact on investors' appropriate risk-return trade-off. Kolm and Ritter (2019) used this utility function as the reward function.

6.3 Other RL algorithms

As mentioned previously, the state space in our method is combinatorial, stochastic, and enormous. We cannot expect to find an optimal policy or optimal value function, even given infinite time, to explore every state. In fact, as the number of episodes increases, the time required increases nonlinearly. For example, in the SARSA method, every step the agent searches the Q-Table for the optimal action. However, after many simulations, the Q-Table can become very large. Therefore, searching for an optimal action can also be time consuming. Q-learning, another TD control method, also faces this problem.

A deep Q-Net (DQN) can solve these problems. In this method, only the network, structure, and parameters of the deep learning network must be stored. Similar inputs result in similar outputs, which means a stronger generalization ability. However, the training results may not converge because of the introduction of nonlinear functions to approximate the Q-Table. In this study, we did not compare different RL algorithms because the focus was on RL itself instead of comparing the efficiency of different algorithms. The introduction here provides a choice when the performance of SARSA is not good.

7 Conclusion

In this article, we propose a simple reinforcement learning method that can train an agent and obtain a hedging strategy that can make the agent gain more rewards than the rewards from the "buy and hold" policy.

Subsequently, an example is provided to demonstrate the power of RL. We introduced a structural product called the Phoenix option. By RL, the Sharpe ratio of investors' portfolios at maturity is higher than when no hedging strategy is adopted.

In existing research on reinforcement learning hedging strategies, the reward function is relatively fixed; for example, Kolm and Ritter (2019) introduced the function property term as a regularizer in the reward function. In fact, in the real investment world, the utility function is not commonly used as a criterion to evaluate good or bad investments, while our work considers this by using the Sharpe ratio as the reward function, which also allows us to further investigate hedging strategies in real markets in the future.

In our method, an explicit pricing formula is unnecessary; however, it can improve the running speed of our algorithm to a certain extent. In addition, we do not need a pre-calculated perfect dynamic hedging strategy, and its existence is not a necessity. The strategy for the initial state of the algorithm is 0. The algorithm does is to solely explore and exploit. When a perfect dynamic hedging strategy does not exist, the proposed method can still obtain a better hedging strategy with training.

Our method can also be modified according to different preferences. In this study, we considered the psychological changes of investors in the investment process, that is, we controlled for the Sharpe ratio in each period. We also considered cases in which investors can focus only on the Sharpe ratio at maturity or give it more weight. Furthermore, other performance measures, such as the Sortino ratio instead of the Sharpe ratio, are also applicable in this model.

The power of RL is significantly higher than that. Compared with other theories used to make decisions in finance, such as the stochastic control theory, RL can fully utilize historical data and even generate more data, and with exploration and exploitation, there can be fewer assumptions and better performance.

Our approach also has some drawbacks in that we do not use real market data because of product complexity. Based on our work, more research can be conducted. In addition to the expansions mentioned in Sect. 5, that is, different weights, performance measures, and RL algorithms, we can use historical data to generate the training and test sets in the presence of transaction costs. Another possible research direction would be to use parallel algorithms to increase training speed.

References

Almahdi S, Yang SY (2017) An adaptive portfolio trading system: a risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. Expert Syst Appl 87:267–279

Andrew AM (1999) reinforcement learning: an introduction by Richard S. Sutton and Andrew G. Barto, Adaptive computation and machine learning series. MIT Press (Bradford Book), Cambridge [1998, ISBN 0-262-19398-1,(hardback,£ 31.95). Robotica, 17(2), 229–235]

Bakshi G, Cao C, Chen Z (1997) Empirical performance of alternative option pricing models. J Financ 52(5):2003–2049

Bao BK, Yin BQ, Xi HS (2008) Infinite-horizon policy-gradient estimation with variable discount factor for Markov decision process. In: 2008 3rd international conference on innovative computing information and control. IEEE, pp 584–584

Barucci E, Cherubini U, & Landi L (1996) No-arbitrage asset pricing with neural networks under stochastic volatility. In: Neural networks in financial engineering: Proceedings of the third international conference on neural networks in the capital markets. World Scientific, New York, pp 3–16

Berger-Tal O, Nathan J, Meron E, Saltz D (2014) The exploration-exploitation dilemma: a multidisciplinary framework. PLoS ONE 9(4):e95693

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

Buehler H, Gonon L, Teichmann J, Wood B, Mohan B, Kochems J (2019b) Deep hedging: hedging derivatives under generic market frictions using reinforcement learning. Swiss Finance Institute Research Paper (19–80)

Caldentey R, Haugh M (2006) Optimal control and hedging of operations in the presence of financial markets. Math Oper Res 31(2):285–304

Cao J, Chen J, Hull J, Poulos Z (2021) Deep hedging of derivatives using reinforcement learning. J Financ Data Sci 3(1):10–27

Das SP, Padhy S (2017) A new hybrid parametric and machine learning model with homogeneity hint for European-style index option pricing. Neural Comput Appl 28(12):4061–4077

Du J, Jin M, Kolm PN, Ritter G, Wang Y, Zhang B (2020) Deep reinforcement learning for option replication and hedging. J Financ Data Sci 2(4):44–57

Dupire B (1994) Pricing with a smile. Risk 7(1):18–20

Fliess M, Join C (2010) Delta hedging in financial engineering: towards a model-free setting. In: 18th Mediterranean conference on control and automation, Marrakech

Gammerman A, Vovk V (2007) Hedging predictions in machine learning. Comput J 50(2):151–163

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

Gu S (2022) Deep reinforcement learning with function properties in mean reversion strategies. J Financ Data Sci 4(4):54–65

Halperin I (2020) Qlbs: Q-learner in the black-scholes (-merton) worlds. J Deriv 28(1):99–122

Hull J, White A (1987) The pricing of options on assets with stochastic volatilities. J Financ 42(2):281–300

Joo J, Moon H (2021) Quantum variational PDE solver with machine learning. arXiv preprint arXiv:2109.09216

Kolm PN, Ritter G (2019) Dynamic replication and hedging: a reinforcement learning approach [J]. J Financ Data Sci 1(1):159–171

Lim YS, Gorse D (2018) Reinforcement learning for high-frequency market making. In: ESANN 2018-Proceedings, European symposium on artificial neural networks, computational intelligence and machine learning. ESANN, pp 521–526

Merton RC (1976) Option pricing when underlying stock returns are discontinuous. J Financ Econ 3(1–2):125–144

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

Pitis S (2019) Rethinking the discount factor in reinforcement learning: a decision theoretic approach. In: Proceedings of the AAAI conference on artificial intelligence, vol 33(01), pp 7949–7956

Sharpe WF (1966) Mutual fund performance. J Bus 39(1):119–138

Shin HJ, Ryu J (2012) A dynamic hedging strategy for option transaction using artificial neural networks. Int J Softw Eng Appl 6(4):111–116

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

Taleb NN (1997) Dynamic hedging: managing vanilla and exotic options, vol 64. Wiley, New York

Vittori E, Trapletti M, Restelli M (2020) Option hedging with risk averse reinforcement learning. In: Proceedings of the first ACM international conference on AI in finance, pp 1–8

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