

Deep Hedging: Continuous Reinforcement Learning for Hedging of General Portfolios across Multiple Risk Aversions

Phillip Murray
J.P. Morgan
London, UK

Ben Wood
J.P. Morgan
London, UK

Hans Buehler
TU Munich
Munich, Germany

Magnus Wiese
TU Kaiserslautern
Kaiserslautern, Germany

Mikko S. Pakkanen
Imperial College London
London, UK

ABSTRACT

We present a method for finding optimal hedging policies for arbitrary initial portfolios and market states. We develop a novel actor-critic algorithm for solving general risk-averse stochastic control problems and use it to learn hedging strategies across multiple risk aversion levels simultaneously. We demonstrate the effectiveness of the approach with a numerical example in a stochastic volatility environment.

CCS CONCEPTS

• **Theory of computation** → **Sequential decision making; Reinforcement learning.**

KEYWORDS

Deep Hedging, Reinforcement Learning, Risk Averse, Transaction Costs

ACM Reference Format:

Phillip Murray, Ben Wood, Hans Buehler, Magnus Wiese, and Mikko S. Pakkanen. 2022. Deep Hedging: Continuous Reinforcement Learning for Hedging of General Portfolios across Multiple Risk Aversions. In *3rd ACM International Conference on AI in Finance (ICAIF '22)*, November 2–4, 2022, New York, NY, USA. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3533271.3561731>

1 INTRODUCTION

Traditionally, financial inventory management of securities and over-the-counter (OTC) derivatives has relied on classic *complete market* models of quantitative finance, such as the seminal work by Black, Scholes and Merton. There, we assume that all risks can be eliminated by cost-free continuous-time trading in markets with infinite depth, whose price processes

evolve undisturbed by our trading activity. The main characteristic of this approach is that we use the sensitivities of the complete-market value of our inventory with respect to changes in market parameters – *the Greeks* – as risk management signals.

However, since the idealized assumptions of a complete market do not apply in real markets, it is not surprising that complete market models require constant manual adjustments and oversight, for example adjusting delta by “skew delta”, smoothing barriers priced with local volatility, and taking into account market impact when trading vega to hedge auto-callable products. Financial inventory management is therefore to this day heavily dependent on human decision making and heuristic adjustments.

This approach made sense when availability of data and computation power was scarce. However, in recent years there has been a growing interest in the application of modern machine learning methods to the pricing and hedging of portfolios of OTC derivatives. In particular, the approach known as *deep hedging* [5] has paved the way to a model-agnostic, data-driven framework for scalable decision making in financial inventory management.

In the deep hedging framework, it is possible to construct a dynamic risk-adjusted hedging strategy in complex hedging instruments under market frictions for a fixed portfolio with a fixed time horizon T using episodic policy search, using market simulation until the maturity of the portfolio. Further iterations of the deep hedging algorithm have been explored, such as applications to rough volatility [17], and American-style options [2].

However, to date, all approaches face the significant limitation that the model must be trained using a fixed portfolio, over a fixed time horizon. Thus, we need to train a new model every time we do a new trade. Furthermore, the model is trained against some risk-averse objective, often with a somewhat arbitrarily chosen risk preference. Hence, in market states where we might want to change our tolerance to risk, such as times of high volatility, we again need to train a new model with this new risk aversion level.

In this article, we address all of the above challenges. Firstly, we learn general policies for arbitrary portfolios by employing a common feature representation of our portfolio

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICAIF '22, November 2–4, 2022, New York, NY, USA

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-9376-8/22/10...\$15.00

<https://doi.org/10.1145/3533271.3561731>

instruments. Specifically, we mimic exactly what a human trader does - we compute the “book value”, risk metrics, and *Greeks* of our portfolio at each market state. This therefore gives us a feature space which is shared across different payoffs, and is also a *Linear Markov Representation (LMR)* in the sense that we can represent linear combinations of portfolio instruments by the corresponding linear combination of their feature vectors. By doing so, we learn a policy which is a mapping from this feature space to actions, which learns the correct state-dependent adjustment to model sensitivities.

Secondly, we formulate deep hedging in a more classical reinforcement learning framework, and derive a *risk-averse* Bellman representation of the deep hedging problem. We then develop a novel actor-critic algorithm for solving this risk-averse problem. Our model-free, on-policy algorithm uses deep function approximators to learn both policies and value functions for general portfolios of instruments. By using monetary utility functions, our value function also represents an indifference price for the portfolio. Hence with the output of our actor-critic algorithm we are able to price and hedge arbitrary portfolios of derivatives.

Thirdly, we show how we can train multiple hedging strategies with different risk aversion levels in parallel, representing them all with a single neural network that learns the mapping from market state and risk aversion level to hedging action. We present this as a general actor-critic algorithm for solving risk-averse reinforcement learning problems across multiple risk aversion levels in Algorithm 1.

There has been some interest in accounting for the variability of returns within the fields of *risk-averse* or *safe* reinforcement learning [3, 13], including approaches such as CVaR optimization [7, 8], and distributional reinforcement learning [9]. So far, these lack applications into the potentially high dimensional continuous state and action spaces considered here. On the other hand, related approaches that apply deep reinforcement learning methods to option hedging and pricing include [11] who solve a simplified version by discretizing the action space, [25] who apply a similar methodology to a mean-variance objective (which has the limitation of not being monotonic with respect to the terminal PnL), and [20] who apply the risk adjustment to the individual rewards rather than the entire return sequence, which can lead to suboptimal hedging strategies. Our approach is the first to formulate the problem in terms of a nonlinear risk-averse Bellman equation and present an algorithm for solving this over continuous state and action spaces, and in particular to do so across multiple risk aversion levels simultaneously.

2 REINFORCEMENT LEARNING

We first introduce the risk-averse reinforcement learning framework, before turning to our specific use-case of hedging portfolios of derivatives.

2.1 Markov decision processes

We consider the problem of sequential decision making of an agent acting within an environment over discrete timesteps

over a finite time horizon. We formulate this as a Markov Decision Process $(\mathcal{S}, \mathcal{A}, r, p)$, where at each timestep, the agent observes the current state $s_t \in \mathcal{S}$, takes an action $a_t \in \mathcal{A}$ based on the current state. Here we consider continuous state and action spaces: $\mathcal{S} \subset \mathbb{R}^d$, and similarly $\mathcal{A} \subset \mathbb{R}^n$. The transition dynamics are given by $p(s_{t+1}|s_t, a_t)$. The environment is in general stochastic, giving rise to the stochastic scalar-valued reward function $r = r(s_t, a_t)$, $s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$. The actions are drawn from a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ which we will assume to be deterministic.

It is standard to define the *returns* from a state s_t as the sum of future rewards from following the policy π

$$R^\pi(s_t) = \sum_{i=t}^T r(s_i, a_i) \quad (1)$$

Similarly, it is standard to define a value function over states as the expected returns of following the policy π starting from state s_t

$$V^\pi(s_t) = \mathbb{E}[R^\pi(s_t)|s_t] \quad (2)$$

The objective of standard reinforcement learning is to learn a policy π which maximizes the expected returns, typically from some distribution over initial states. In this formulation, only the expectation is taken into consideration, not the variance or higher moments of the distribution of future rewards. For this reason, we may describe this objective as *risk neutral*. A range of on-policy and off-policy algorithms exist, but all approaches to solving this optimization make use of some variation on the linear Bellman equation

$$V^\pi(s_t) = \mathbb{E}[r(s_t, a_t^\pi) + V^\pi(s_{t+1})] \quad (3)$$

and subsequently perform either value iteration, policy iteration or a combination of the two in actor-critic methods. See [15, 19, 23] for an overview of approaches.

2.2 Risk-averse reinforcement learning

When managing the risk of a portfolio of derivatives however, the inherent stochastic nature of the system makes the variability of future cashflows of paramount importance, and an optimization criterion which accounts for both risk and reward is essential.

To this end, we will not use expected returns, but a risk-adjusted return measure to assess the value of future rewards. We will assess our policy with respect to the certainty equivalent of the *exponential utility* which is given for any random variable X as

$$U_\lambda(X) = -\frac{1}{\lambda} \log \mathbb{E}[\exp(-\lambda X)] \quad (4)$$

for a risk-aversion parameter $\lambda \in \mathbb{R}^+$. The exponential utility has the limits $U(X) = \mathbb{E}[X]$ for $\lambda \rightarrow 0$ (risk-neutral) and $U(X) = \inf X$ for $\lambda \rightarrow \infty$ (worst-case). It is an example of a *monetary utility* as it is increasing (more is better), concave (risk-averse) and cash-invariant, in the sense that for a fixed amount $c \in \mathbb{R}$, we have $U(X + c) = U(X) + c$. This last property implies that $U_\lambda(X - U_\lambda(X)) = 0$, which

means we may interpret $-U_\lambda(X)$ as the *monetary utility equivalent* of X – that is, the amount of cash we would need to add to the portfolio to make it acceptable for that level of risk aversion. The negative of the exponential utility is an example of a *convex risk measure* [12], other examples include CVaR which has been explored in the risk-sensitive RL literature [7, 8, 24], but has the somewhat unnatural property of *coherence*: $U(nX) = nU(X)$ suggesting that the risk value of a position grows linearly with position size, whereas superlinear growth is more appropriate.

The exponential utility has the Taylor expansion $U_\lambda(X) = \mathbb{E}[X] - \lambda/2\text{Var}[X] + \mathcal{O}(\lambda^2)$ and indeed when X is normally distributed, it becomes equivalent to the classic Markowitz mean-variance portfolio objective.

Applying the exponential utility to the MDP setting, we may define the value of a state now as the utility of the future returns from that state

$$V^\pi(s_t) = U_\lambda\left(\sum_{i=t}^T r(s_i, a_i)\right) \quad (5)$$

and the goal is to find a policy π which maximizes this objective. In the original deep hedging approach, this was done relative to a fixed initial state s_0 , and solved via periodic policy search [5]. Here we present a more general approach that is able to deal with a variety of initial states. It can be shown that the value function defined by 5 satisfies a non-linear *risk-averse* Bellman equation [10]

$$\begin{aligned} V^\pi(s_t) &= U_\lambda(r(s_t, a_t) + V^\pi(s_{t+1})) \\ &= -\frac{1}{\lambda} \log \mathbb{E}_{s_{t+1}|s_t, a_t} [\exp(-\lambda(r(s_t, a_t) + V^\pi(s_{t+1})))] \end{aligned} \quad (6)$$

The exponential utility is the only utility apart from the simple expectation to satisfy this property of *time consistency* [14, 21].

2.3 Multiple levels of risk aversion

The above formulation of the value function allows for sensitivity to risk, but leaves open the question of what a sensible level of risk aversion is. For low values of λ , our value function behaves almost like the expectation, and learned policies will be almost risk-neutral. On the other hand, for high λ , policies may be overcautious due to extreme sensitivity to large losses. Rather than choosing a specific level, we can consider an ensemble of agents, each with different risk aversion levels, where we want to learn an optimal policy for each agent. We can learn across these multiple agents in parallel, by including λ in the “agent state”, hence the policy and the value function become implicitly dependent on the risk aversion level, i.e. $V^\pi(s_t; \lambda)$ indicates the value of following policy π from the state s_t , given that we apply the risk aversion λ in the utility of returns. We drop any explicit dependence on λ and still write $V^\pi(s_t)$ for brevity.

We will consider risk aversion to be a static property of the agent which is independent of the environment state, and

therefore does not change over time. Extensions to the case where the risk aversion is a function of the state $\lambda_t = \lambda(s_t)$ are possible, but we would lose the time consistency of the objective, and are not considered here.

Unlike the standard case of risk neutral reinforcement learning, it is not immediately clear how to use the risk-averse Bellman equation to perform policy or value updates. For the policy, note that the objective of $\sup_\pi V^\pi(s_t)$ is equivalent to solving the exponentiated objective

$$\inf_\pi \frac{1}{\lambda} \exp(-\lambda V^\pi(s_t)) \quad (8)$$

Similarly, we can reformulate the Bellman equation for the value function as

$$\exp(-\lambda V^\pi(s_t)) = \mathbb{E}_{s_{t+1}|s_t, a_t} [\exp(-\lambda(r(s_t, a_t) + V^\pi(s_{t+1})))] \quad (9)$$

Using these simple reformulations, we can develop an algorithm for solving the risk-averse MDP via actor-critic methods.

2.4 Risk-averse actor-critic algorithm

We now propose an actor-critic algorithm for solving this risk-averse MDP. We represent our policy and value functions with neural network function approximators, π^θ and V^ϕ respectively. By doing so, we can represent all policies across different levels of risk aversion by a single network, where the risk aversion level is included as a feature to the network, and do the same for the value function.

Then we can update the policy by following the gradient of the exponentiated objective with our approximated value function, i.e.

$$\mathbb{E}_{s_t} \left[\mathbb{E}_{s_{t+1}|s_t, a_t} \left[\frac{1}{\lambda} \exp(-\lambda(r_t + V^\phi(s_{t+1}))) \right] \right] \quad (10)$$

where the outer expectation is over visited states s_t , and the inner is over transitions from each state. We may unbiasedly estimate the inner expectation using single transitions from our current policy, so that the policy network is updated with respect to the loss

$$\mathcal{L}(\theta) = \mathbb{E}_{s_t} \left[\frac{1}{\lambda} \exp(-\lambda(r_t + V^\phi(s_{t+1}))) \right] \quad (11)$$

For the critic, define the target as

$$y_t = r_t + V^\phi(s_{t+1}) \quad (12)$$

then a natural candidate for the critic loss would be the squared error between the exponentiated value and the exponentiated target: $\mathbb{E}[(e^{-\lambda y_t} - e^{-\lambda V^\phi(s_t)})^2]$. However, in practice this is observed to suffer from numerical instability – any large, negative state value estimates can cause numerical issues, and the loss is notably asymmetric, i.e. for most states the loss can be decreased by simply increasing the estimates of the values of both the current and next state, irrespective of the observed reward, whilst also making the gradient close to zero. We are therefore relying on the corrective updates from end of episode value estimates where we have some

ground truth such as $V(s_T) = 0$. This generally leads to very noisy and unstable updates. Instead, we construct a loss function which smooths the value function updates, and penalizes value function overestimates as well as underestimates, while still targeting the same exponentiated Bellman equation. So consider the function

$$f(V) = \mathbb{E}_{s_{t+1}|s_t, a_t} \left[\frac{1}{\lambda} \exp(-\lambda(y_t - V(s_t))) - V(s_t) \right] \quad (13)$$

The minimizer of this function satisfies

$$\mathbb{E}_{s_{t+1}|s_t, a_t} [\exp(-\lambda(y_t - V(s_t))) - 1] = 0 \quad (14)$$

which in turn leads to the Bellman equation described in Equation 9. Therefore, in a similar nature to the actor loss, we may use the following loss for the critic

$$\mathcal{L}(\phi) = \mathbb{E}_{s_t} \left[\frac{1}{\lambda} \exp(-\lambda(y_t - V^\phi(s_t))) - V^\phi(s_t) \right] \quad (15)$$

where again the expectation is over the states and the inner conditional expectation has been estimated using single transitions. While the target y_t depends on ϕ , we follow the standard approach of ignoring this dependency and only updating the value function with the gradients from $V^\phi(s_t)$. To do this, we use a copy of the network which we denote as the target network $\bar{V}$ which we do not train and copy the parameters of V^ϕ over to this target. To stabilize the estimates of the target values, we use a soft update rule where the parameters described in [22], which slowly transfers the weights via $\bar{\phi} = \tau\phi + (1 - \tau)\bar{\phi}$ for some small τ .

By iteratively updating the policy and value functions on minibatches of experience sampled from the policy transitioning through the environment we obtain an actor-critic algorithm for the risk-averse MDP. We learn across risk aversion levels by specifying a set of levels of interest – for example an interval $[\lambda_{min}, \lambda_{max}]$ – and then sample from those at the beginning of each episode, assigning a different level to each trajectory in the minibatch. Since risk aversion tends to work on a logarithmic scale, we found that a good choice was sampling uniformly in log space. We summarize the full algorithm applied to episodes in Algorithm 1.

Note that (after some constant scaling and shifting), the loss for our critic is $\mathbb{E}[(y_t - V^\phi(s_t))^2] + \mathcal{O}(\lambda)$ so that in the limit $\lambda \rightarrow 0$ we recover the standard squared (linear) Bellman error used in risk-neutral algorithms. Similarly, the loss for our actor becomes $\mathbb{E}[-r(s_t, a_t) - V^\phi(s_{t+1})]$, so that in the limit $\lambda \rightarrow 0$, our algorithm will learn the risk neutral policy.

3 DEEP HEDGING

We now focus on the primary application of our risk-averse actor-critic algorithm to solve the problem of hedging arbitrary portfolios of derivatives with respect to the exponential utility.

Algorithm 1 Multiple risk aversion actor-critic

Initialize actor network π and critic network V with weights θ, ϕ
Initialize target critic network $\bar{V}$ with weights $\bar{\phi} = \phi$
for Episode $1, \dots, M$ **do**
 Sample a batch of N initial states s_0 and risk aversion levels λ
 for $t = 0, \dots, T$ **do**
 Select actions from current policy $a_t = \pi(s_t)$
 Execute actions, receive rewards r_t and transition into new states s_{t+1}
 Compute the target

$$y_t = r_t + \bar{V}(s_{t+1})$$

Update the critic by gradient descent with respect to the loss

$$\mathcal{L}(\phi) = \frac{1}{N} \sum_{i=1}^N \frac{1}{\lambda} \exp(-\lambda(y_t - V(s_t))) - V(s_t)$$

Update the actor by gradient descent with respect to the loss

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N \frac{1}{\lambda} \exp(-\lambda(r_t + V(s_{t+1})))$$

Update target critic network $\bar{\phi} = \tau\phi + (1 - \tau)\bar{\phi}$
end for
end for

3.1 Market environment

We consider a setting of trading in a financial market over discrete time steps. We denote by m_t the market state today. The market state contains all information available to us today such as current market prices, time, past prices, bid/asks, social media feeds and so forth.

We assume that the agent is a trader who is in charge of a *portfolio* – also known as a “book” – of financial instruments such as securities, OTC derivatives or currencies, which we will denote by z_t , and the space of such portfolios $\mathcal{Z}$. This portfolio changes over time due to the changes in the market state, and as a result of the trader’s activity. In order to risk manage her portfolio, the trader has access to n liquid hedging instruments h_t such as forwards, vanillas options, swaps. Across different market states they will usually not be the contractually same fixed-strike, fixed-maturity instruments: instead, they will usually be defined relative the prevailing market in terms of time-to-maturities and strikes relative to at-the-money. See [5] for details.

At each timestep the trader may trade in $a_t \in \mathbb{R}^n$ units of the liquid hedging instruments, which then become part of the portfolio at the next time step as follows. Denote by z'_{t+1} the current portfolio z_t at the next time step (i.e.

as if no action were taken), and h'_{t+1} the state at the next time step of the instruments available to trade at the current time step. Then the portfolio at the next time step will be $z_{t+1} = z'_{t+1} + a_t \cdot h'_{t+1}$.

Trading incurs costs, which in general can depend on both the current portfolio and the market. This allows modelling trading restrictions based on our current position such as short-sell constraints, or restrictions based on risk exposure. We represent this by a transaction cost function $c(s, a)$ which is normalized to $c(s, 0) = 0$, non-negative, and convex, giving rise to a convex set of admissible actions $\mathcal{A}(s) = \{a \in \mathbb{R}^n : c(s, a) < \infty\}$. A simple example of such a cost function is proportional transaction cost $c(s_t, a_t) = \alpha |a_t| |h_t|$.

We assume that $p(m_{t+1}|s_t, a_t) = p(m_{t+1}|m_t)$ – that is, actions have no market impact, and the future state of the market only depends on the current market state.

At every time step, the portfolio produces random cashflows depending on the market state, which we will denote $r(z_t)$, and the trader must pay for any instruments traded, giving rise to the reward

$$r(s_t, a_t) = r(z_t) - a_t \cdot h_t - c(s_t, a_t) \quad (16)$$

It is common practice for human traders to consider the *book value* $B(z)$ of their portfolio, which we define to be its official mark-to-market, computed using the prevailing market data m . For liquidly traded instruments such as vanillas, this could be a simple closing price or a weighted mid-price. For exotics, this may be the result of computing more complex derivative model prices. We could choose to include the change in book value of the portfolio $B(z_{t+1}) - B(z_t)$ inside the reward function, which has the benefit of reducing both the delay and sparsity of rewards – for example, a 1M vanilla option has at most one cashflow, which is multiple timesteps after the action of trading it. Therefore by including the change in book value we could provide more reward signals to our agent. In this implementation, we choose to only use realized cashflows in the reward. However, we use the book value to address any positions which are still outstanding at the end of the episode. At time T we value the portfolio by its book value – or equivalently, we enforce liquidation of the remaining portfolio at book value, so that $r_T = B(z_T)$. For example, if we have accumulated a position of δ units of an underlying S , then the book value at time T could be $\delta S_T - c(s_T, \delta)$ – the proceeds from liquidating the position at the current midprice.

3.2 Representing portfolios

The state can therefore be written as $s_t = (m_t, z_t)$. But as mentioned above, we need a representation of z_t such that hedging instruments we have just traded can be added into our existing portfolio in a meaningful way. For this, we define a feature map $\Phi_t : \mathcal{Z} \rightarrow \mathbb{R}^d$ that maps any portfolio into a feature vector, $\Phi_t(z) = (\Phi_t^1(z), \dots, \Phi_t^d(z))$. Moreover, let Φ_t be a *Linear Markov* feature map, in the sense that the mapping of the combined portfolio $z_{t+1} = z'_{t+1} + a_t \cdot h'_{t+1}$ is given by

$$\Phi_{t+1}(z_{t+1}) = \Phi_{t+1}(z'_{t+1}) + a_t \cdot \Phi_{t+1}(h'_{t+1}) \quad (17)$$

We then call $\Phi_t(z)$ a *Linear Markov representation (LMR)* of the portfolio. To construct such a feature map, we follow the common practice of human traders of computing risk sensitivities – or *Greeks* – of our portfolio instruments. The Greeks represent various sensitivities of the value of a derivative to a change in model parameters on which the financial instrument is dependent. For example, the delta of an instrument z written on an underlying S is defined as $\Delta = \partial z / \partial S$. Thus, we specify some model, and compute vectors of sensitivities for our portfolio and hedging instruments and use them to define the feature map Φ_t . Note that the map changes at each step, since the sensitivities are computed with respect to the current model parameters, which are themselves conditional on the market state m_t . Importantly, we do not assume that the model used to compute the sensitivities perfectly reflects the true dynamics of the environment, only that there is sufficient information in the representation to learn policies from. The LMR can also include the book value and any other risk metrics which are linear in nature.

In addition, if $\mathcal{Z}$ represents some subset of portfolios with common, product-specific features, we can optionally include these in our feature representation, such as strikes, fixing dates and so on. These will not be part of the LMR but additional features passed to the networks.

3.3 Actor-critic deep hedging

Having set up the reward function and state representation, we can apply Algorithm 1 to learn hedging strategies for arbitrary portfolios across a range of risk aversion levels. Using a market simulator, such as described in [26], we can randomly sample a batch of initial market states and portfolio states $s_0 = (m_0, z_0)$. Then we can transition through the simulated market, trading in the available hedging instruments and receiving rewards according to 16, and following the algorithm, update our actor and critic at each transition via stochastic gradient descent. Once trained, we will be able to hedge arbitrary portfolios that were seen in training, and from the monetary utility property, the value function enables us to price any portfolio, given the current market state.

4 NUMERICAL RESULTS

In this section we present some results in a simulated stochastic volatility environment to highlight the generality of our approach. For this purpose, we assume that our market follows the dynamics of the Heston [16] model, although we stress that this is only for illustration and our algorithm is model independent. we begin with a brief reminder of the Heston model.

4.1 Heston market environment

Consider a market with a single underlying asset, with options written on that asset. The market dynamics under the Heston model are specified by the stochastic differential equations

$$\begin{aligned} dS_t &= \mu S_t dt + \sqrt{v_t} S_t dB_t \\ dv_t &= \kappa(\theta - v_t)dt + \xi \sqrt{v_t} dW_t \end{aligned}$$

where B and W are Brownian motions with correlation $\rho \in [-1, 1]$, S is the liquidly tradeable asset with drift μ , and v is the stochastic variance process of S , which follows a mean reverting Cox-Ingersoll-Ross process, with mean reversion rate κ , mean reversion level θ , and volatility of volatility ξ . Sample paths of S are simulated using the Broadie-Kaya scheme [1, 4]. We generate 200,000 sample paths with parameters $\rho = -0.7, \mu = 0, \kappa = 8, \theta = 0.00625, \xi = 1$, with daily data over a finite time horizon of $T = 30$ days and normalized to $S_0 = \$1$. The choice of $\mu = 0$ ensures that we have no *statistical arbitrage* and so the policy is purely hedging [6]. For each path we randomly assign a short position in a call option with strike $K \in \{0.9, 0.925, \dots, 1.1\}$ expiring at T , with fixed notional of \$100, so that the payoff is $-100(S_T - K)^+$. For sample efficiency we keep the maturity of the portfolios constant, to avoid episodes where many trading steps are taken on an empty portfolio after the option has expired. We use proportional transaction costs of 0.002.

For the LMR, we compute Black-Scholes Greeks for the portfolios, using the square root of the variance process as the implied volatility. Strictly speaking, this is unobserved, but the rationale is that we imagine a case where we are able to observe it through some other liquidly traded variance products. Furthermore, we do not compute Greeks with respect to the true, unknown, Heston model, but with respect to the Black-Scholes model, hence the feature vector is an imperfect representation of the state, in particular because the Black-Scholes model does not account for the stochasticity of volatility. The Greeks used for the LMR are delta (Δ), gamma (Γ), vega (V), theta (Θ), vanna, charm, and vomma. These can all be computed via closed-form formulas in the Black-Scholes model. In addition to the LMR we add portfolio-specific features of strike and time-to-maturity.

The use of the Greeks to represent our portfolio state presents the challenge of feature scaling – for example the gamma of a close-to-expiry option can be orders of magnitude higher than its delta. Unlike in supervised learning, features cannot simply be rescaled in the naive way since we do not know a priori what the distribution of states under the optimal policy will be. Solutions to this problem have been explored, such as batch normalization [18], which employs dynamic feature scaling, using running means and standard deviations of the states. Instead of this, we take a policy which we know to be reasonably close to the optimal policy and compute the state mean and standard deviations under this policy for feature normalization. In this case, we can do so by running the Black-Scholes delta hedging policy on the environment $\pi^{BS}(s_t) = -\Delta(s_t)$. We will also use this policy as a benchmark for performance comparison.

For the actor, we use a three layer feedforward neural network with 256 units in each layer, and sigmoid activations. We use a skip connection, so that the output of the network is

$\pi(s_t) = -\Delta(s_t) + \pi^\theta(s_t)$, which can be interpreted as learning the correct adjustment to the Black-Scholes delta hedge.

For the critic, we also use a three layer feedforward neural network with 256 units in each layer, and sigmoid activations. A reasonable first order approximation of the state value should be the book value of our current portfolio, which is computed under the same Black-Scholes model for the options, and taken as the observed market midprice of the underlying. Hence, our value function outputs $V(s_t) = B(s_t) + V^\phi(s_t)$. This can be seen as making the correct risk adjustment to the model implied cash value of our current portfolio. For the soft target updates we used $\tau = 0.001$.

At the beginning of each episode, we sample a batch of risk aversions uniformly on a log scale from the interval $[10^{-4}, 1]$, and the risk aversion for each trajectory is held fixed through the episode. Thus the actor network will learn hedging policies across these risk aversion levels.

The learning rates were 1×10^{-3} for the actor and 1×10^{-4} for the critic. We train with minibatch sizes of 2048 for 100,000 episodes.

4.2 Assessing the value function

To assess the performance of the critic, we check that it correctly matches the true realized utility by generating 100 random portfolios, with strike sampled uniformly from $[0.9, 1.1]$ and λ sampled uniformly from $[10^{-4}, 1]$. We then compute the true value of each portfolio by running the policy on a sample of 10,000 paths and then computing the utility of the terminal hedged PnL. We then compute the RMSE between value function estimates and true utility, giving a value of $RMSE = 0.014$ indicating that our value function is able to almost perfectly reproduce the correct utility.

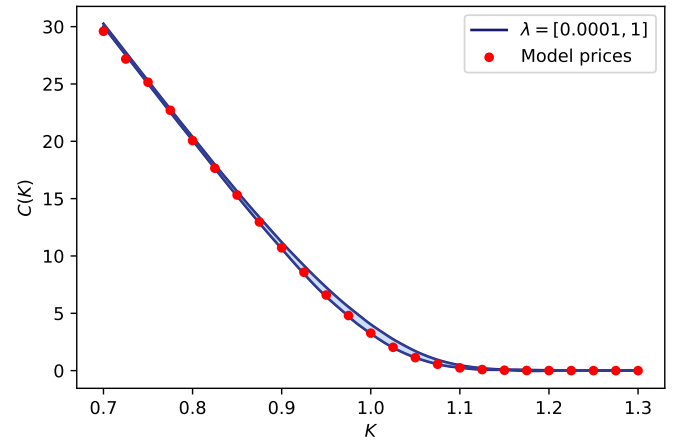


Figure 1: Call prices obtained by the value function for maturity $T = 30$ compared with risk-neutral model prices. Low λ produces prices very close to the model price, and higher λ results in additional risk premium added.

Next, we check that the value assigned to the portfolio corresponds correctly to the risk-adjusted price, as implied by the monetary utility. In Figure 1 we compare the prices of 30-day call options over a range of risk aversions. As $\lambda \rightarrow 0$ the value function correctly approximates the risk neutral model price for low λ , and since $U_\lambda(X) \leq \mathbb{E}[X]$, as λ increases we charge an additional risk premium to make the trade acceptable. Note that to get this correspondence it was essential that the market was free from statistical arbitrage, in which the value of the empty portfolio is zero. We also achieve very good extrapolation performance, pricing strikes outside of the range $[0.9, 1.1]$ seen in the training data with high degree of accuracy.

Finally, in Figure 2 we price options for maturities in the range 1 – 60 days. We observe that the value function extrapolates well, preserving the convexity in strike and monotonicity in maturity, despite no explicit constraints.

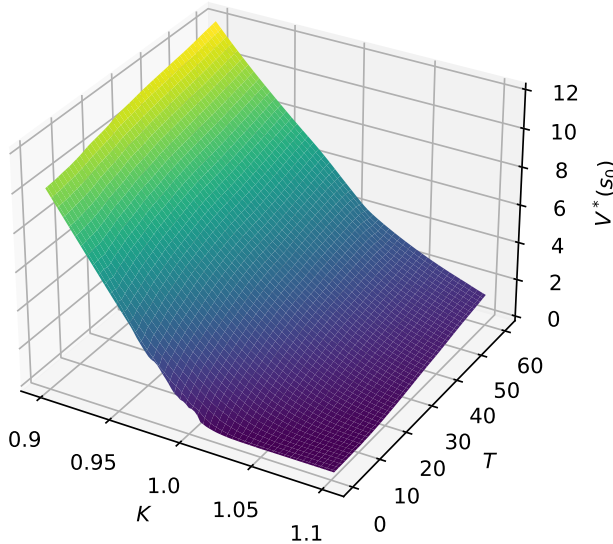


Figure 2: Value function estimates of call prices for maturities between 1–60 days. Maturities from 31–60 were never seen in training.

4.3 Assessing the policy

We now assess the performance of the hedging policy. Figure 3 shows the initial hedging action as a function of strike and risk aversion. For low λ , the policy leaves all positions unhedged. As we increase the risk aversion, the level of hedging smoothly increases. For deep out of the money options, the agent prefers to leave it unhedged even at higher risk aversion levels. Due to market frictions, the value of the unhedged position is higher than the hedged one.

Figure 4 compares the distribution of terminal hedged PnL of the policy at different risk aversion levels, showing

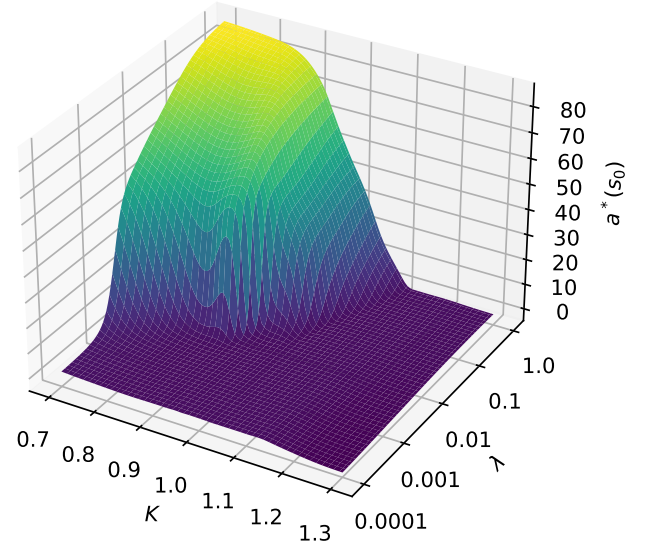


Figure 3: Initial hedge as a function of strike and risk aversion. The policy network smoothly interpolates from zero position for low λ to an almost full hedge for high λ .

tightening of the distribution as risk aversion increases. Note also that by the cash invariance of the value function, each distribution is centred around the (negative) corresponding risk adjusted price.

In Table 1 we compare our actor-critic method with the baselines of Black-Scholes delta hedging and vanilla deep hedging across a range of strikes for $\lambda = 0.1$. For vanilla deep hedging we train a new model for each payoff and then test the utility of all policies on a test set of 20,000 new paths. The vanilla deep hedging model used the same network architecture as the policy network and trained for 1,000 epochs. The policy reproduces the utility of vanilla deep hedging, which is in all cases better than delta hedging. The key point here is that although the performance is equivalent, or perhaps slightly worse, than the vanilla deep hedging model, we use one single actor critic model for all instruments. The training time for actor-critic is of a similar order to each individual vanilla deep hedging model (1000 epochs being roughly equivalent to 100,000 episodes), therefore we observe a linear runtime improvement with the number of instruments to hedge.

To illustrate the effect of dynamically changing the risk aversion level, we test the policy on an environment with a position in a 1.1 strike call. π_1 starts with $\lambda = 0.005$ but then increases to $\lambda = 0.1$ at $t = 5$ and then $\lambda = 1$ at $t = 20$, π_2 has $\lambda = 0.005$ for all t , and π_3 starts with $\lambda = 0.01$, then decreases to $\lambda = 10^{-4}$ at $t = 15$. Figure 5 shows the median and quartiles of the hedging positions as we evaluate on the environment. Note that the median appears approximately

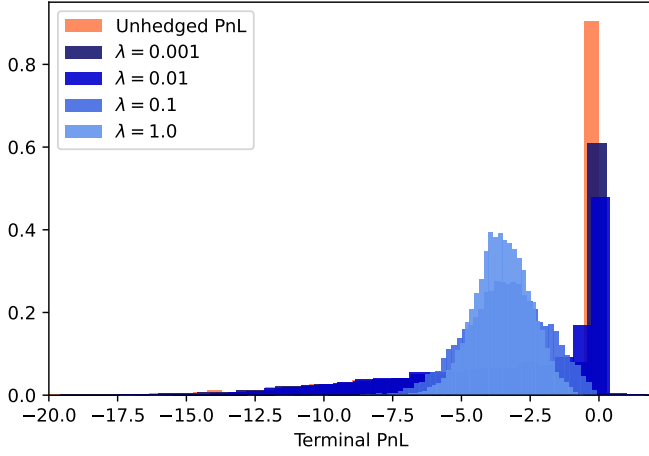


Figure 4: Terminal hedged PnL of a 30-day at the money call option for varying risk aversion levels.

Strike	Black-Scholes delta hedging	Vanilla deep hedging	Actor-critic deep hedging
0.9	-11.10	-10.87	-10.91
0.95	-7.06	-6.74	-6.68
1.0	-3.78	-3.36	-3.41
1.05	-1.52	-1.14	-1.11
1.1	-0.43	-0.26	-0.27

Table 1: Utility of hedged PnL against strike, for the Black-Scholes delta hedging baseline, vanilla deep hedging and our actor-critic approach.

constant, but on a pathwise basis, the hedges are indeed dynamically changing as the state changes. Each time the risk aversion increases, the policy instantly increases its hedge, but when the risk aversion drops, it prefers to just keep the position until maturity, to avoid paying transaction costs.

5 CONCLUSION

In this paper, we proposed an actor-critic algorithm for solving risk-averse stochastic control problems in continuous state and action spaces, across multiple risk aversion levels, using neural networks to approximate both the policy and a value function representing an optimized certainty of the exponential utility. With this algorithm, we were able to learn prices and hedging strategies for a range portfolios of derivatives across multiple risk aversion levels in a market with stochastic volatility. We utilised a Linear Markov representation of the portfolio state, and our policy was able to learn the correct state-dependent adaptations to model parameters to give an optimal hedging policy. This paper therefore lays the groundwork for a full “Bellman” deep hedging, and future work extending to the infinite time horizon setting and expanding

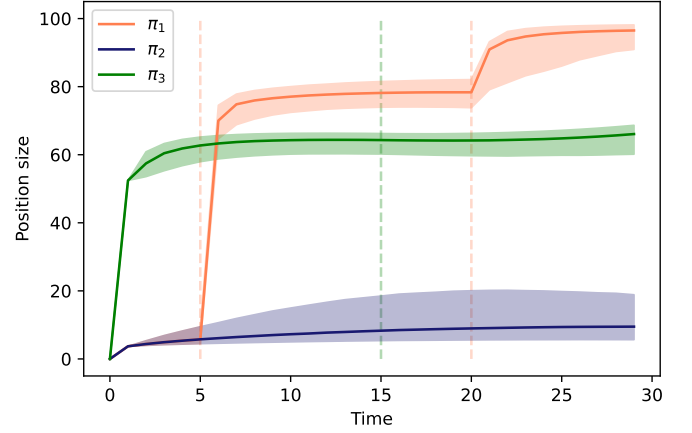


Figure 5: Plot of median and quartiles of hedge positions under dynamically adjusted risk aversion level. Under policy π_1 the agent increases its risk aversion at $t = 5$ and again at $t = 20$, both times leading to an instant increase in the hedge position.

the numerical implementation to a broader class of exotics and hedging instruments will be explored.

REFERENCES

- [1] Leif BG Andersen, Peter Jäckel, and Christian Kahl. 2010. Simulation of square-root processes. *Encyclopedia of Quantitative Finance* (2010), 1642–1649.
- [2] Sebastian Becker, Patrick Cheridito, and Arnulf Jentzen. 2020. Pricing and hedging American-style options with deep learning. *Journal of Risk and Financial Management* 13, 7 (2020), 158.
- [3] Lorenzo Bisi. 2022. Algorithms for risk-averse reinforcement learning. (2022).
- [4] Mark Broadie and Özgür Kaya. 2006. Exact simulation of stochastic volatility and other affine jump diffusion processes. *Operations research* 54, 2 (2006), 217–231.
- [5] H. Buehler, L. Gonon, J. Teichmann, and B. Wood. 2019. Deep hedging. *Quantitative Finance* (2019), 1–21. <https://doi.org/10.1080/14697688.2019.1571683> arXiv:<https://doi.org/10.1080/14697688.2019.1571683>
- [6] Hans Buehler, Phillip Murray, Mikko S. Pakkanen, and Ben Wood. March 2022. Deep hedging: Learning to remove the drift. *Risk* (March 2022).
- [7] Yinlam Chow and Mohammad Ghavamzadeh. 2014. Algorithms for CVaR optimization in MDPs. *Advances in neural information processing systems* 27 (2014).
- [8] Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. 2015. Risk-sensitive and robust decision-making: a cvar optimization approach. *Advances in neural information processing systems* 28 (2015).
- [9] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. 2018. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
- [10] Oscar Dowson, David P Morton, and Bernardo K Pagnoncelli. 2020. Multistage stochastic programs with the entropic risk measure. *Preprint* (2020).
- [11] Jiayi Du, Muyang Jin, Petter N Kolm, Gordon Ritter, Yixuan Wang, and Bofei Zhang. 2020. Deep reinforcement learning for option replication and hedging. *The Journal of Financial Data Science* 2, 4 (2020), 44–57.
- [12] Hans Föllmer and Alexander Schied. 2010. Convex and coherent risk measures. *Encyclopedia of Quantitative Finance* (2010), 355–363.

- [13] Javier Garcia and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.
- [14] Michael Gimelfarb, André Barreto, Scott Sanner, and Chi-Guhn Lee. 2021. Risk-Aware Transfer in Reinforcement Learning using Successor Features. *Advances in Neural Information Processing Systems* 34 (2021), 17298–17310.
- [15] Ben Hambly, Renyuan Xu, and Huining Yang. 2021. Recent advances in reinforcement learning in finance. *arXiv preprint arXiv:2112.04553* (2021).
- [16] Steven L Heston. 1993. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The review of financial studies* 6, 2 (1993), 327–343.
- [17] Blanka Horvath, Josef Teichmann, and Žan Žurič. 2021. Deep hedging under rough volatility. *Risks* 9, 7 (2021), 138.
- [18] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 448–456.
- [19] Sebastian Jaimungal. 2022. Reinforcement learning and stochastic optimisation. *Finance and Stochastics* 26, 1 (2022), 103–129.
- [20] Petter N Kolm and Gordon Ritter. 2019. Dynamic replication and hedging: A reinforcement learning approach. *The Journal of Financial Data Science* 1, 1 (2019), 159–171.
- [21] Michael Kupper and Walter Schachermayer. 2009. Representation results for law invariant time consistent functions. *Mathematics and Financial Economics* 2, 3 (2009), 189–210.
- [22] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).
- [23] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (2 ed.). MIT Press, Cambridge.
- [24] Aviv Tamar, Yinlam Chow, Mohammad Ghavamzadeh, and Shie Mannor. 2016. Sequential decision making with coherent risk. *IEEE transactions on automatic control* 62, 7 (2016), 3323–3338.
- [25] Edoardo Vittori, Michele Trapletti, and Marcello Restelli. 2020. Option hedging with risk averse reinforcement learning. In *Proceedings of the First ACM International Conference on AI in Finance*. 1–8.
- [26] Magnus Wiese, Ben Wood, Alexandre Pachoud, Ralf Korn, Hans Buehler, Phillip Murray, and Lianjun Bai. 2021. Multi-asset spot and option market simulation. *arXiv preprint arXiv:2112.06823* (2021).

DISCLAIMER

Opinions and estimates constitute our judgement as of the date of this Material, are for informational purposes only and are subject to change without notice. It is not a research report and is not intended as such. Past performance is not indicative of future results. This Material is not the product of J.P. Morgan’s Research Department and therefore, has not been prepared in accordance with legal requirements to promote the independence of research, including but not limited to, the prohibition on the dealing ahead of the dissemination of investment research. This Material is not intended as research, a recommendation, advice, offer or solicitation for the purchase or sale of any financial product or service, or to be used in any way for evaluating the merits of participating in any transaction. Please consult your own advisors regarding legal, tax, accounting or any other aspects including suitability implications for your particular circumstances. J.P. Morgan disclaims any responsibility or liability whatsoever for the quality, accuracy or completeness of the information herein, and for any reliance on, or use of this material in any way. Important disclosures at: www.jpmorgan.com/disclosures.