Empirical Deep Hedging

Prof Juho Kanniainen
A joint work with Mr Oskari Mikkilä

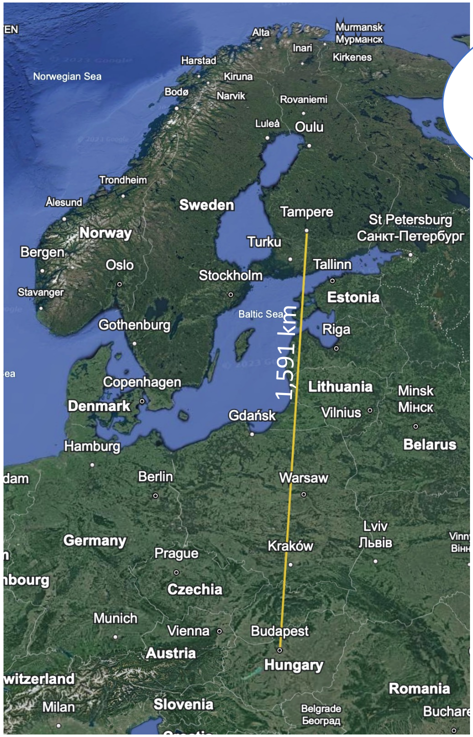
Financial Computing and Data Analytics Group,
Dept of Computing Sciences,
Tampere University, Finland
juho.kanniainen@tuni.fi

This talk is based on paper
Mikkilä and Kanniainen (2023), "Empirical Deep Hedging", *Quantitative Finance*, 23(1), 111-122.

Codes are available at `https://github.com/oskarimikkila/Empirical-Deep-Hedging`

7th Workshop on Understanding the Diversity of Financial Risk
Machine Learning for Prediction and Optimisation
Budapest, Hungary
November 24, 2023

Where I'm from



Did you know that the Finnish and Hungarian languages have common roots?
The Times: Tampere, the fairytale Finnish city you've probably never heard of
https://www.thetimes.co.uk/article/the-fairytale-finnish-city-you-ve-probably-never-heard-of-57gtr6v9v

Our research: ML & HFT

- We have actively researched the prediction of the mid-price of stocks using high-frequency limit order book data.
- The data used has been the Nasdaq ITCH feed, from which the states of the order books have been reconstructed.
- The models can be used for trading, market making, market surveillance etc.
- Example papers:
  - Shabani, M., Tran, D.T., Kanniainen, J. and Iosifidis, A., 2023. Augmented bilinear network for incremental multi-stock time-series classification. *Pattern Recognition* (IF 7.196).
  - Tran, D.T., Iosifidis, A., Kanniainen, J. and Gabbouj, M., 2018. Temporal attention-augmented bilinear network for financial time-series data analysis. *IEEE Transactions on Neural Networks and Learning Systems* (IF 14.255).
  - Mäkinen, Y., Kanniainen, J., Gabbouj, M. and Iosifidis, A., 2019. Forecasting jump arrivals in stock prices: new attention-based network architecture using limit order book data. *Quantitative Finance* (IF 2.222).

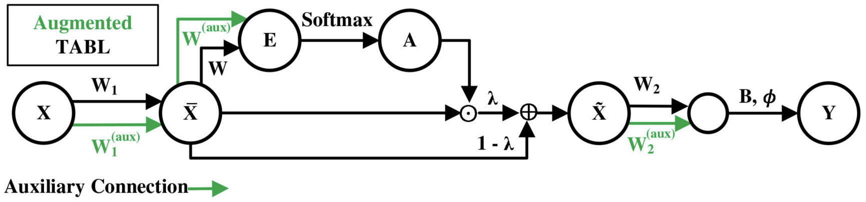
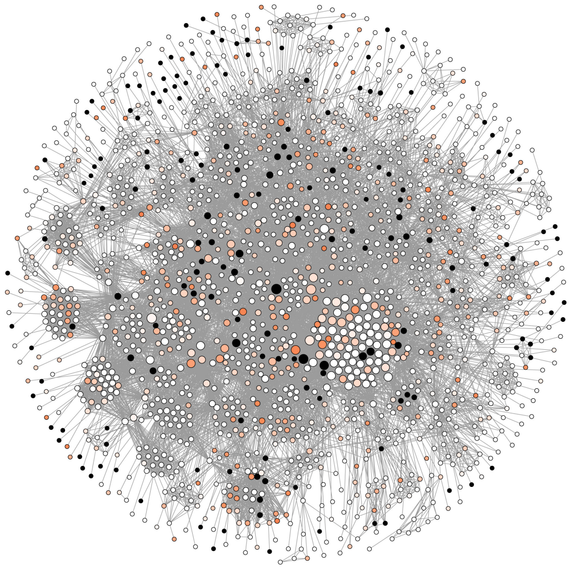


Figure. Architecture of Augmented bilinear network

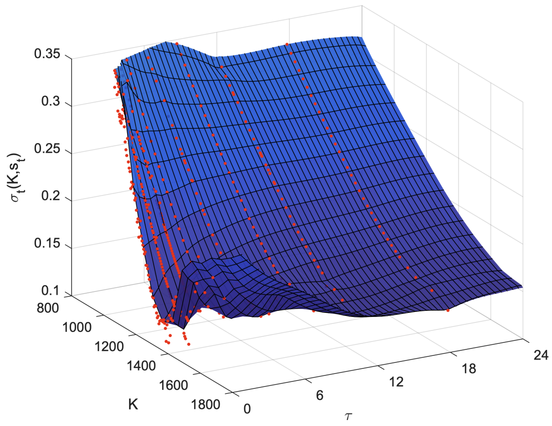
Investor Network Research

- In our investor network research, we analyze how investors transform private information between each other and utilize it in the stock market.
- We have developed methods to identify information networks in stock markets (see Baltakienė et al. 2021).
- Additionally, we have introduced a graph neural networks-based tool to rank investors according to their suspiciousness regarding the use of insider information (see Baltakys et al 2023).
- Example papers:
  - Baltakienė, M., Kanniainen, J. and Baltakys, K., 2021. Identification of information networks in stock markets. *Journal of Economic Dynamics and Control* (IF 1.9).
  - Baltakys, K., Baltakienė, M., Heidari, N., Iosifidis, A. and Kanniainen, J., 2023. Predicting the trading behavior of socially connected investors: Graph neural network approach with implications to market surveillance. *Expert Systems with Applications* (IF 8.5).



EMPIRICAL OPTION PRICING

- Here, the focus has been on analyzing the empirical performance of alternative volatility and jump models.
- Moreover, recently we have applied Reinforcement Learning to the hedging of index options. This direction appears very promising, offering a number of topics for future research.
- Example papers:
  - Kanniainen, J., Lin, B. and Yang, H., 2014. Estimating and using GARCH models with VIX data for option valuation. *Journal of Banking & Finance* (IF 3.7).
  - Yang, H. and Kanniainen, J., 2017. Jump and volatility dynamics for the S&P 500: Evidence for infinite-activity jumps with non-affine volatility dynamics from stock and option markets. *Review of Finance* (IF 5.059).
  - Mikkilä, O. and Kanniainen, J., 2023. Empirical deep hedging. *Quantitative Finance* (IF 2.222).



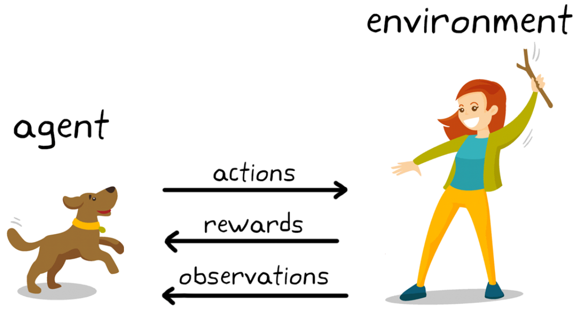
Motivation for Hedging with Reinforcement Learning

- Under stochastic volatility with two sources of uncertainty in the model, delta hedging alone is insufficient to eliminate all risk.
- In option hedging, a tradeoff exists between minimizing the variance of wealth increments and the transaction costs.
- The hedging problem can be naturally expressed using Reinforcement Learning (RL): Observing the state of the environment necessitates taking actions to adjust the number of underlying shares held in the portfolio, which, in turn, directly impacts the state.
- It is not surprising that considerable attention has been devoted to hedging options using RL, as evidenced by several studies (Kolm and Ritter 2019; Buehler et al. 2018; Kolm and Ritter 2020; Cao et al. 2021; Halperin 2019; Halperin 2020; Du et al. 2020; Giurca and Borovkova 2021).
- Thus far, research papers have trained agents using synthetic data generated by specific volatility and/or jump models.
- We demonstrate **how to successfully train RL agents purely through data-driven methods using empirical option data**, without prior specification of the underlying volatility or jump processes.

Deep Reinforcement Learning

Basics

- RL systems have two entities that interact with each other: an **agent** and an **environment**.
- The optimal action of an agent is defined as an action that **maximizes the expected lifetime reward** in light of the observations on the current environment.
- Once the reward and other settings are determined by a user, the machine employs trial and error to iteratively develop an optimal solution to the problem.
- Perhaps the most important innovation in RL has been the **combination of RL and deep neural networks** to capture the optimal policies.



Source of the figure:
`https://se.mathworks.com/discovery/reinforcement-learning.html`

Deep Reinforcement Learning

Basics

- At each time step, an agent observes the **state of the environment**, $s_t \in \mathcal{S}$, and then selects an **action** $a_t \in \mathcal{A}$ with respect to its deterministic **policy** $\mu : \mathcal{S} \to \mathcal{A}$ with parameter vector $\phi \in \mathbb{R}^n$.
- After taking action a_t, the agent receives a reward $r_{t+1} \in \mathbb{R}$, determined by a **reward function** $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and observes a new state of the environment s_{t+1}.
- Then based on s_{t+1}, an agent takes a new action a_{t+1}, receiving reward r_{t+2} and so on.
- As a result, we have a trajectory $s_0, a_0, r_1, s_1, a_1, r_2, \ldots$ This continues until a terminal state is reached.
- The period between the start and terminal states is called an **episode**.
- In this paper, the episode is five trading days long with 35 time stamps (length of the step is one trading our).
- The problem is modelled as a Markov Decision Process (MDP) with a stationary transition dynamics distribution with conditional density $p(s_{t+1}|s_1, a_1, \ldots, s_t, a_t) = p(s_{t+1}|s_t, a_t)$ (history does not matter).
- **Discrete actions** have a known, limited space of available actions.
- **Continuous actions** might have upper and lower limits, but any action in between can be chosen.
- Therefore, continuous actions are in line with the assumption of frictionless in delta hedging, for which reason this research is based on the continuous action space.

Deep Reinforcement Learning
Reward

- The total cumulative discounted **reward** from time-step t onwards is defined by

$$\tilde{r}_t = \sum_{i=0}^{\infty} r_{t+i}\gamma^i,$$

where $r_{t+i} = R(s_{t+i-1}, a_{t+i-1})$ is the immediate reward received at time $t+i$ and γ is the **discount parameter** with $0 < \gamma \leq 1$.

- Agent's goal is to find the **optimal policy** μ with parameters ϕ to **maximize the expected cumulative reward** over the entire episode, which can be written as

$$\begin{aligned} J(\mu_\phi) &= \int_{\mathcal{S}} \rho^{\mu_\phi}(s;t)R(s,a)ds \\ &= \mathbb{E}_{s\sim\rho^{\mu_\phi}}\left[\tilde{r}_t|\mu_\phi\right], \end{aligned}$$

where $a = \mu(s|\phi)$ and ρ^{μ_ϕ} is the **discount state distribution**,

$$\rho^{\mu_\phi}(s';t) := \int_{\mathcal{S}} \sum_{i=0}^{\infty} \gamma^{t+i} p_t(s)p(s \to s'; t+i, \mu_\phi)ds,$$

where s' is a state after transitioning for $t+i$ time steps from state s with probability $p(s \to s'; t+i, \mu_\phi)$ and μ_ϕ is the **policy function** with parameters ϕ.

Deep Reinforcement Learning

Learning procedure

- In **Q-learning** (Watkins and Dayan 1992), the goodness of an action is evaluated in terms of the expected reward.
- Given that $\tilde{r}_t = \sum_{i=0}^{\infty} r_{t+i}\gamma^i$, let us define the value of action a in state s under policy μ_ϕ as the expected reward:

$$\begin{aligned} Q^{\mu_\phi}(s,a) &= \mathbb{E}_{s\sim\rho^{\mu_\phi}}\left[\tilde{r}_t | s_t = s, a_t = a; \mu_\phi\right] \\ &= r(s_t,a_t) + \gamma\mathbb{E}_{s\sim\rho^{\mu_\phi}}\left[Q^{\mu_\phi}\left(s_{t+1}, \mu(s_{t+1}|\phi)\right)\right], \end{aligned} \tag{1}$$

  where a is determined by $\mu_\phi(s)$.
- Following Mnih et al. (2013), in deep Q-learning, the **optimal action-value function**

$$Q(s,a|\theta) \approx Q^*(s,a) = \max_\phi Q^{\mu_\phi}(s, \mu(s|\phi))$$

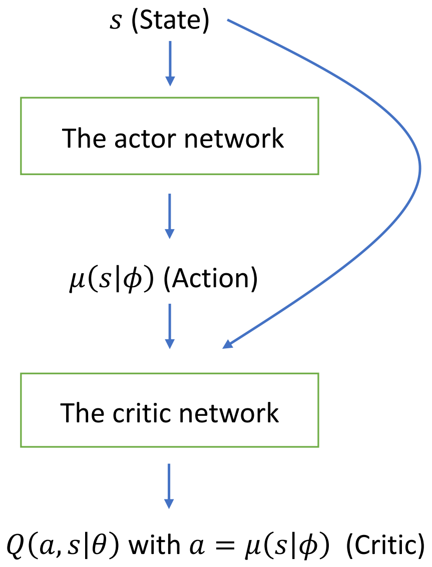
  is approximated by (deep) neural networks.
- If the action space is continuous, enumerating all possible actions is impractical due to their infinite number. This challenge is addressed by employing function approximators with two neural networks:
  - **actor** $\mu(s|\phi)$ (used for selecting actions)
  - **critic** $Q(s,a|\theta)$ (criticizes/evaluates the actions taken by the actor)

  with parameters ϕ and θ, respectively.

Deep Reinforcement Learning

Learning procedure

- Multiple algorithms have been introduced to estimate the action-value function $Q^{\mu_\phi}(s, a)$.
- In continuous action space research, perhaps the best known deterministic policy gradient algorithm is the Deep Deterministic Policy Gradients (DDPG) by Lillicrap et al. (2015).
- DDPG is known as the equivalent of Deep Q-Learning for continuous action space.
- The idea is that the **actor** directly maps states to actions instead of outputting the probability distribution across a discrete action space.
- For a given actor, the expected rewards depends only on the environment.
- DDPG is **off-policy**, meaning that the agent learns from experiences that may have been generated from a different policy than the one it is currently optimizing.
- The policy $\mu(s|\phi)$ can be updated through the **deterministic policy gradient algorithm** (DPG).



LEARNING PROCEDURE

- Function approximator is optimized by minimizing

$$L(\theta_i) = \mathbb{E}_{s_i, a_i, r_i, s_{i+1}} \left(Q(s_i, a_i | \theta_i) - y_i \right)^2 \tag{2}$$

  with respect to θ, where

$$y_i = r(s_i, a_i) + \gamma Q(s_{i+1}, a_i^* | \theta_i^*), \tag{3}$$

  is the **target**. Here $a_i^* = \mu(s|\phi^*) + \epsilon$ is the **target action**, where ϵ is exploration noise.
- That is, we create **target versions** for both the actor and critic neural network models, $Q(s, a|\theta^*)$ and $\mu(s|\phi^*)$, to calculate the target values.
- The primary role of $Q(s, a|\theta^*)$ is to estimate the future value of actions taken in the next state (not the current state where the immediate reward $r(s_i, a_i)$ is received.
- **Temporal difference parameters** θ_i^* and ϕ_i^* are used to compute the target at iteration $i + 1$.
- The main critic network estimates the Q-value (action-value) for the current state and action, while the target critic network estimates the Q-value for the next state and the action chosen by the target actor network.

Deep Reinforcement Learning

Learning procedure

- In Monte Carlo methods, value estimates are updated only at the end of an episode, based on the total accumulated reward. Temporal Difference (TD) Learning, on the other hand, updates value estimates based on other, already learned estimates, and it does so after each step, not waiting for the episode to finish.
- Imagine a simple game where an agent moves through a given sequence of states, say $(S_1, S_2, S_3, \dots S_N)$ towards a goal. The agent receives a reward only upon reaching the final state S_N. With TD,
  - The agent moves from S_1 to S_2. It doesn't know the final outcome yet, but it updates the value of S_1 based on its estimate of S_2's value and the reward received (if any) from moving to S_2.
  - This process repeats at each step. When moving from S_2 to S_3, the value of S_2 is updated based on the estimated value of S_3.
  - Essentially, the value of each state is updated by looking ahead one step, using the value estimate of the next state.
- In this paper, we use **Twin Delayed Deep Deterministic Policy Gradients, TD3**, introduced by Fujimoto, Hoof, and Meger (2018). TD3 can be seen as an extension of DDPG.
- The advantage of TD3 is that it minimizes the effect of overestimation bias using two critic networks to mitigate, taking the minimum of their Q-value estimates.
- It has six neural networks in total: $1\times$ actor, $2\times$ critic and own target networks for each three.

DEEP REINFORCEMENT LEARNING

LEARNING PROCEDURE

Algorithm 1: TD3 Algorithm (Source Fujimoto, Hoof, and Meger 2018)

```
Result: Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$, and actor network $\pi_\phi$ with random parameters $\theta_1, \theta_2, \phi$
1  Initialize target networks $\theta_1^* \leftarrow \theta_1, \theta_2^* \leftarrow \theta_2, \phi^* \leftarrow \phi$
2  Initialize replay buffer $\mathcal{B}$
3  for $t = 1$ to $T$ do
4      Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward $r$ and new
         state $s'$
5      Store transition tuple $(s, a, r, s')$ in $\mathcal{B}$
6      Sample mini-batch of $N$ transitions $(s, a, r, s')$ from $\mathcal{B}$
7      $a^* \leftarrow \pi_{\phi^*}(s') + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma^*), -c, c)$
8      $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i^*}(s', a^*)$
9      Update critics $\theta_i \leftarrow \arg\min_{\theta_i} \frac{1}{N} \sum (y - Q_{\theta_i}(s, a))^2$
10     if $t \mod d = 0$ then
11         Update $\phi$ by the deterministic policy gradient:
12         $\nabla_\phi J(\phi) = \frac{1}{N} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
13         Update target networks:
14         $\theta_i^* \leftarrow \tau\theta_i + (1 - \tau)\theta_i^*$
15         $\phi^* \leftarrow \tau\phi + (1 - \tau)\phi^*$
16     end
17 end
```

Deep Reinforcement Learning for Hedging

Earlier research

- Kolm and Ritter (2019) and Du et al. (2020) use methods with discrete action space. Additionally, Halperin (2020) considers both discrete-space and continuous-space versions but with (non-deep) Q-learning without the use of neural networks
- There are also papers working on continuous action domain, such as (Cao et al. 2021) and (Giurca and Borovkova 2021) using DDPG.
- However, when testing simpler methods than TD3, including DDPG, we observed that they either failed to converge or began to learn and then later diverged.
- At the same time, the stabilizing features introduced in TD3 appeared to render it the most viable method.

Deep Reinforcement Learning for Hedging

Reward function

- To derive the reward function, we follow Cao et al. (2021) to maximize
$$\mathbb{E}(w_T) - \xi \mathbb{SD}(w_T),$$
with $\xi > 0$, where w_T is the wealth at time T.
- Kolm and Ritter (2019) and Du et al. (2020) use a nearly identical objective function, but they consider variance instead of standard deviation.
- The reward r_t should be a function of wealth increments, thus
$$\mathbb{E}(r_t) = \mathbb{E}(\Delta w_t) - \xi \mathbb{SD}(\Delta w_t).$$
- At each step, we approximate the one-period reward function as
$$r_t = \mathrm{PnL}_t - \xi \left| \mathrm{PnL}_t \right|, \tag{4}$$
where PnL_t (Profit-and-Loss) is the change in agent's wealth between time steps:
$$\mathrm{PnL}_t = H_t^O \left(C_t - C_{t-1} \right) + H_t^S \left(S_t - S_{t-1} \right) - c \left| S_t \left(H_t^S - H_{t-1}^S \right) \right|. \tag{5}$$
- Here H_t^O is the option position, which equals -1 (+1) for a short (long) call option position, C_t is the price of the option, H_t^S represents the holdings in the underlying asset at time t, being positive (negative) for a short (long) call option position, S_t is the price of the underlying asset at time t, and c denotes the transaction costs.
- There is a trade-off between transaction costs (c) and the risk caused by insufficient rebalancing (the standard deviation of the P&L).

Deep Reinforcement Learning for Hedging

States

- Actions are determined by the state of the environment, which is captured by the following variables:
  - Moneyness
  - Time to maturity
  - Current number of shares held
  - Implied volatility
- The state *does not* need to include the option Greeks (the neural networks can infer them from the state variables).
- Moreover, to learn a policy independent of any pricing model and to investigate if Deep Reinforcement Learning (DRL) can independently learn a hedging policy, it is important to exclude Greeks.
- As neural networks are universal function approximators, the Black-Scholes delta hedging can be considered as a special case within our model.
- We implement z-score normalization for the input variables.

Deep Reinforcement Learning for Hedging
Settings

- We set the hedging interval to 60 minutes (equals seven re-balances a day).
- Episode length will be set to 35 time-steps, i.e. five trading days.
- For TD3 hyper parameters, $\tau = 0.001$ controls how fast the target networks are updated.
- The NNs have three layers, each of size 250 (the adding of the 3rd layer provided marginal improvement).
- For the activation functions of the hidden layers, we use Leaky ReLU with $\alpha = 0.05$:

$$h(i) = \begin{cases} i & \text{if } i > 0 \\ \alpha \times i & \text{otherwise.} \end{cases}$$

- The activation function for the actor's output layer is hyperbolic tangent with outputs of a range of $[-1, 1]$, which then later scaled outside the neural network to a range of $[0, 1]$.
- For the critic network, the output layer outputs a single value with linear activation.
- Together with a small learning rate (1E–04) and a high batch size of 10,000 a stabilizing effect is achieved.
- Adam is used as an optimizer.

Data

- We use S&P 500 index intraday options data from Jan 3rd, 2006 to Dec. 31st, 2013 (spanning eight years, 2009 trading days), provided by CBOE Livevol.
- The option roots considered are those used by CBOE in the calculation of the VIX index.
- Given the hedging interval of 60 minutes, hourly observations are utilized.
- We assume that the underlying asset can be traded in any fractional amount.
- Transaction costs are assumed to be 1 BPS. If transaction costs were large, the DRL agent would outperform the classic Black-Scholes delta hedging strategy "too easily", as the DRL agent can strategically avoid transaction costs.
- The training data includes options with moneyness $0.83 \leq S/K \leq 1.17$ and options set to expire within 10–90 days, each considered over a 5-day episode.
- For empirical validation and testing, the target strikes are set at $S/K \in [0.85, 0.925, 1, 1.075, 1.15]$ and target maturity times are 10, 30, and 60 days. Options closest to these target strikes and maturities (in percentage terms) are selected, and then utilized for a 5-day episode.

Data

Table. Snapshot of option data.

Time	Code	Underlying	Strike	Days to mat.	Rate	Call price	Call bid	Put price	Put bid
2006-Jan-03 09:31	SPX	1253.00	800	18	0.0438	454.45	454.2	0.1	0
2006-Jan-03 09:31	SPX	1253.00	825	18	0.0438	429.45	429.2	0.1	0
...			...						
2006-Jan-03 09:31	SXZ	1253.00	1400	18	0.0438	0.1	0	144.25	144
2006-Jan-03 09:31	SXM	1253.00	1500	18	0.0438	0.1	0	244.05	243.8
2006-Jan-03 09:31	SPX	1253.00	800	46	0.0456	455.05	454.8	0.1	0
2006-Jan-03 09:31	SPX	1253.00	850	46	0.0456	405.35	405.1	0.1	0
...				...					
2006-Jan-03 09:31	SPB	1253.00	1600	347	0.0478	0.65	0.55	297.85	297.6
2006-Jan-03 09:32	SPX	1253.72	800	18	0.0438	455.15	454.9	0.1	0
...									
2006-Jan-03 16:00	SPB	1268.80	1600	347	0.0478	0.75	0.5	284.4	283.4

Table. Summary statistics on the number of unique option contracts and five-day paths for training, validation, and test data.

	Training data (2006–2011)	Validation data (2012)	Test data (2013)
Unique options	7,228	1,539	1,683
Unique five-day paths	12,298	2,703	2,914

Results

Monte Carlo experiments

- In the Monte Carlo experiment, we generate synthetic data from both a **constant volatility** model and a Heston **stochastic volatility** model, which are used to train the DRL agent.
- The parameter estimates for the Heston model in a given simulation were randomly selected from the daily calibrations in 2012.
- To assess the hedging performance of the agent and the benchmarks with out-of-sample data, we consider four measures from the tests:
  - (i) The mean episode P&L is the average of the total P&L over each five-day episode;
  - (ii) The episode P&L standard deviation is the standard deviation of the episode total P&Ls over the 10,000 tests;
  - (iii) Mean episode transaction costs over each five-day episode;
  - (iv) Average rewards accumulated, as determined in Eq. (4).
- As described by Eq. (4), a greater ξ places more emphasis on minimizing risk (the standard deviation of wealth). We report results with $\xi = 1, 2, 3$.

Results

Monte Carlo experiments

Table. Summary statistics on the estimated parameters obtained from daily calibrations of Heston 1993 model in 2012. There were totally 249 trading days in 2012.

	θ	κ	η	ρ	v_0
Min	0.02	0.07	0.04	-1.00	0.01
Max	0.25	9.49	1.00	-0.45	0.08
Median	0.04	3.08	0.56	-0.78	0.02
Average	0.04	3.41	0.62	-0.79	0.03
Standard Deviation	0.02	1.60	0.31	0.16	0.01

Equations used in the Heston model calibration for 2012:

$$\begin{aligned} dS_t &= \mu S_t dt + \sqrt{v_t} S_t dW_{1,t}, \\ dv_t &= \kappa(v_t - \theta)dt + \eta\sqrt{v_t} dW_{2,t}, \end{aligned}$$

where $\kappa > 0, \theta > 0, \eta > 0$, and $dW_{1,t}dW_{2,t} = \rho dt$, $\rho \in [-1, 1]$.

Results

Monte Carlo experiments

Table. Deep reinforcement learning (DRL) agent's performance and hedging cost against the classic Black-Scholes delta hedging benchmark under *constant volatility*. The episode spans 5 days, and the option is a call option expiring in two weeks. The hedge is rebalanced seven times daily. The reported results on P&L and costs are scaled by the underlying price at the beginning of the periods. Panels A–C report the results for $\xi = 1, 2, 3$.

	Mean episode P&L	Std episode P&L	Mean episode transaction costs	Rewards
		Panel A: $\xi = 1$		
Black-Scholes Delta Hedging	-0.0119 %	0.0426 %	-0.0094 %	**8.79**
DRL Agent	-0.0124 %	0.0459 %	-0.0094 %	8.60
		Panel B: $\xi = 2$		
Black-Scholes Delta Hedging	-0.0114 %	0.0436 %	-0.0095 %	**7.17**
DRL Agent	-0.0107 %	0.0490 %	-0.0094 %	6.74
		Panel C: $\xi = 3$		
Black-Scholes Delta Hedging	-0.0108 %	0.0428 %	-0.0094 %	**5.59**
DRL Agent	-0.0112 %	0.0478 %	-0.0094 %	4.90

Results

Monte Carlo experiments

Table. Deep reinforcement learning (DRL) agent's performance and hedging cost against the classic Black-Scholes delta hedging benchmark under Heston *stochastic volatility*. The episode spans 5 days, and the option is a call option expiring in two weeks. The hedge is rebalanced seven times daily. The reported results on P&L and costs are scaled by the underlying price at the beginning of the periods. Panels A–C report the results for $\xi = 1, 2, 3$.

	Mean episode P&L	Std episode P&L	Mean episode transaction costs	Rewards
		Panel A: $\xi = 1$		
Black-Scholes Delta Hedging	-0.0086 %	0.2628 %	-0.0084 %	0.42
DRL Agent	-0.0072 %	0.2343 %	-0.0077 %	**1.66**
		Panel B: $\xi = 2$		
Black-Scholes Delta Hedging	-0.0149 %	0.2551 %	-0.0084 %	-9.19
DRL Agent	-0.0157 %	0.2263 %	-0.0080 %	**-6.82**
		Panel C: $\xi = 3$		
Black-Scholes Delta Hedging	-0.0104 %	0.2606 %	-0.0084 %	-19.44
DRL Agent	-0.0106 %	0.2301 %	-0.0079 %	**-15.97**

When volatility is stochastic, the DRL agent trained with the synthetic data from the Heston model clearly outperforms the classic Black-Scholes delta hedging.

Results

Empirical performance

We evaluate our empirical performance of DRL models that are trained by two different ways:

(i) We train the DRL agent using synthetic data from the calibrated Heston model and test the trained agent with empirical option data (this model is used for Monte-Carlo experiment).
   - The advantage is that there can be as many episodes as needed.
   - Importantly, the policy an agent learns is dependent on the volatility model used.

(ii) We train and test the DRL agent using just **empirical stock and option price observations** instead of the data generated from a model.
   - Here an agent is **trained without any assumptions about volatility dynamics**.
   - To our best knowledge, this is the first research that **trains the agent in a completely data-driven way**.
   - Totally, there are 12,298 unique 5-day periods for training, which was achieved by using different strikes and expiry days for the same time period.
   - This amount of data is found to be sufficient with TD3, which uses experience replay (random samples of previous transitions).
   - By that way, a tuple (s_i, a_i, r_i, s_{i+1}) can be used more than once by random sampling and it addresses the problem of a high correlation between consecutive steps in the environment making the neural network more robust.

Results

Empirical performance

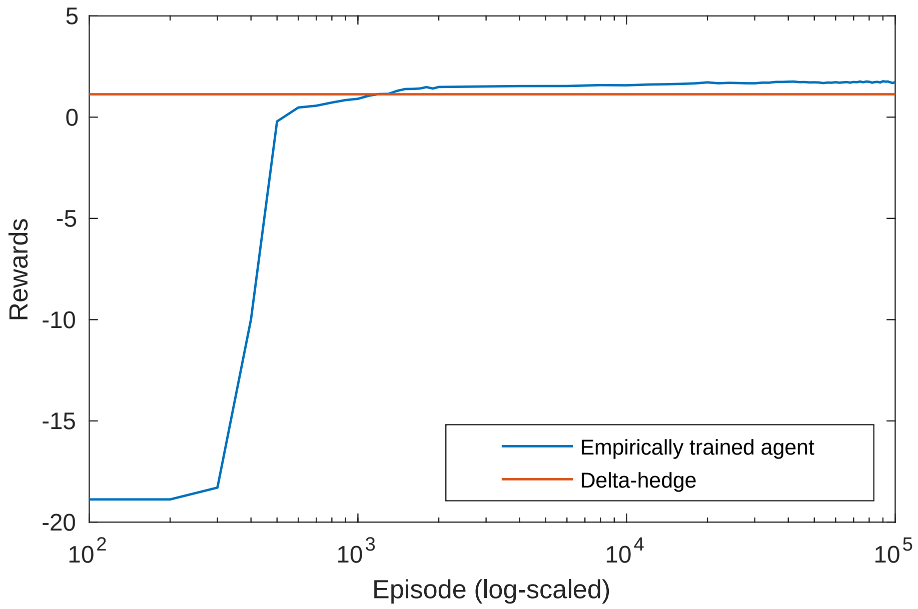


Figure. The performance (rewards) of empirically trained agent against the validation data over the episodes with $\xi = 1$. The performance is compared against the classic Black-Scholes delta hedging strategy over. The performance is plotted against the log-scaled episodes.

Results

Empirical performance

Table. Empirical performance of DRL agents. The hedge period is 5 days. Hedge is rebalanced seven times in a day. Results are reported for different levels of the risk-return tradeoff parameter ξ. The reported results on P&L and costs are scaled by the underlying price at the beginning of the periods.

	Mean episode P&L	Std episode P&L	Mean episode transaction costs	Rewards
		Panel A: $\xi = 1$		
Black-Scholes Delta Hedging	-0.0195 %	0.1515 %	-0.0076 %	4.204
Sim-to-Real DRL Agent trained with Heston model	-0.0095 %	0.1502 %	-0.0068 %	4.375
DRL Agent trained with empirical data	-0.0123 %	0.1415 %	-0.0072 %	**4.466**
		Panel B: $\xi = 2$		
Black-Scholes Delta Hedging	-0.0195 %	0.1515 %	-0.0076 %	-1.896
Sim-to-Real DRL Agent trained with Heston model	-0.0140 %	0.1483 %	-0.0072 %	-1.423
DRL Agent trained with empirical data	-0.0112 %	0.1378 %	-0.0071 %	**-1.361**
		Panel C: $\xi = 3$		
Black-Scholes Delta Hedging	-0.0195 %	0.1515 %	-0.0076 %	-7.997
Sim-to-Real DRL Agent trained with Heston model	-0.0122 %	0.1499 %	-0.0075 %	-7.386
DRL Agent trained with empirical data	-0.0141 %	0.1402 %	-0.0072 %	**-7.272**

RESULTS

EMPIRICAL PERFORMANCE

Main findings:

- **Empirically trained agent shows superior performance**
  - The empirically trained DRL agent outperforms not only the classic Black-Scholes delta hedging, but also the DRL agent trained by synthetic data generated by the calibrated Heston model.
  - The result is robust with all the values of $\xi \in \{1, 2, 3\}$ (risk-return tradeoff).
  - **This implies that DRL can capture the dynamics of S&P 500 from the actual intra-day data and to self-learn how to hedge actual options efficiently** without the prior specification of the underlying volatility or jump processes.
- The DRL agent trained with the synthetic Heston data yields always yields higher returns compared to the classic Black-Scholes delta hedging.
  - This is a very encouraging result, because it suggests that **practitioners could start to implement RL-based techniques to transfer the agent's policy from simulation to real markets** even if there is no sufficient amount of empirical data available to train the models.
  - This finding is consistent with the evidence provided by Giurca and Borovkova (2021), who conclude that the use of RL is suitable for traders taking real-life hedging decisions when the agents are trained on synthetic data.

Future research

- In this study, the hedging period was 60 minutes, but it could be irregular and decided by the agent itself.
- The agent's decision-making process could be structured in two phases: Initially, a decision is made on whether the portfolio should be updated; following that, if an update is decided, the next step involves determining the quantity (and if it buy or sell transaction).
- In addition, the agent could potentially identify mispricing of options and exploit an arbitrage:
  - The agent monitors mispricing: Buy options in response to underpricing and sell options in response to overpricing.
  - Once buy/sell decisions are made, the agent dynamically hedges the option position, utilizing insights from the current research.
  - The agent has the flexibility to hold the option position until maturity **or** unwind it when profitable mispricing is detected again, but in the opposite direction than initially.
  - Specifically, the agent sells options held in a long position upon observing transient overpricing and closes a short position when underpricing is identified.

Thank you!

References I

- Buehler, Hans et al. (2018). **“Deep Hedging”.** In: *Quantitative Finance* 19.8, pp. 1271–1291.
- Cao, Jay et al. (2021). **“Deep hedging of derivatives using reinforcement learning”.** In: *The Journal of Financial Data Science* 3.1, pp. 10–27.
- Du, Jiayi et al. (2020). **“Deep Reinforcement Learning for Option Replication and Hedging”.** In: *The Journal of Financial Data Science* 2.4, pp. 44–57.
- Fujimoto, Scott, Herke Hoof, and David Meger (2018). **“Addressing function approximation error in actor-critic methods”.** In: *International Conference on Machine Learning*. PMLR, pp. 1587–1596.
- Giurca, Alexandru and Svetlana Borovkova (2021). **“Delta Hedging of Derivatives using Deep Reinforcement Learning”.** In: *Available at SSRN 3847272*.
- Halperin, Igor (2019). **“The QLBS Q-Learner goes NuQLear: fitted Q iteration, inverse RL, and option portfolios”.** In: *Quantitative Finance* 19.9, pp. 1543–1553.
- — (2020). **“Qlbs: Q-learner in the black-scholes (-merton) worlds”.** In: *The Journal of Derivatives* 28.1, pp. 99–122.
- Heston, L. Steven (1993). **“A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options”.** In: *The Review of Financial Studies* 6.2, pp. 327–343.
- Kolm, Petter N and Gordon Ritter (2019). **“Dynamic replication and hedging: A reinforcement learning approach”.** In: *The Journal of Financial Data Science* 1.1, pp. 159–171.

REFERENCES II

- Kolm, Petter N and Gordon Ritter (2020). **"Modern perspectives on reinforcement learning in finance".** In: *Modern Perspectives on Reinforcement Learning in Finance (September 6, 2019). The Journal of Machine Learning in Finance* 1.1.
- Lillicrap, Timothy P et al. (2015). **"Continuous control with deep reinforcement learning".** In: *arXiv preprint arXiv:1509.02971*.
- Mikkilä, Oskari and Juho Kanniainen (2023). **"Empirical deep hedging".** In: *Quantitative Finance* 23.1, pp. 111–122.
- Mnih, Volodymyr et al. (2013). **"Playing Atari with Deep Reinforcement Learning".** In: *CoRR*. URL: `http://arxiv.org/abs/1312.5602`.
- Watkins, Christopher JCH and Peter Dayan (1992). **"Q-learning".** In: *Machine learning* 8.3-4, pp. 279–292.