Option Portfolio Replication and Hedging in Deep Reinforcement Learning

Bofei Zhang, Jiayi Du, Yixuan Wang, Muyang Jin

Abstract

Delta hedging is an options strategy that utilizes delta aiming to reduce the risk associated with price movements in the underlying asset while minimizing trading costs, by taking offsetting long or short positions. We employed Deep Reinforcement Learning (DRL) to address this hedging problem in a realistic setting, including discrete time trading with high level of market friction. First, we implemented a simulation environment that simulated stock movements and option pricing by OpenAI Gym. Second, we utilized multiple DRL methods including Deep Q-Learning (DQL), DQL with Preserve Output Precisely and Adaptive Scaling Target (Pop-art) implementation, and Proximal Policy Optimization (PPO) to build agents that can learn how to optimally hedge an option. Third, we evaluated the agent performance in terms of accumulative reward, volatility, trading cost, and profit and loss attribution of our agents and the baseline Delta Hedging policy. We are able to show that PPO has the best performance among all other DRL algorithms. Moreover, PPO has significantly shorter training time and generates more financially sensible policy than other DRL methods.

Finance Background

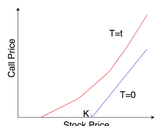
An option is a form of financial derivative that gives the buyer the right to buy or sell a number of corresponding underlying assets at a set price point on a set future date.

Our benchmark strategy, delta hedging, is derived by the well celebrated Black-Scholes Merton (BSM) model, where the pricing of a European call pricing follows the equation:

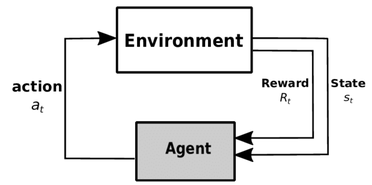
$$c = S_0 N(d_1) - K e^{-rT} N(d_2)$$

$$d_1 = \frac{ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$



We will solve the problem in a more realistic setting where we perform discrete hedging with transaction cost in a market with friction.



Our environment simulates the stock movement and computes option prices according to BSM. At each time step, the agent observes the state, i.e. a 3d vector:

(stock price, time to maturity, number of stocks)

Then, the agent receives a reward that correlated to the wealth gain of this time step:

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

DQL

DQL utilized MLP to approximate the state-action value function by learn a Q function. All Q functions obey Bellman equation:

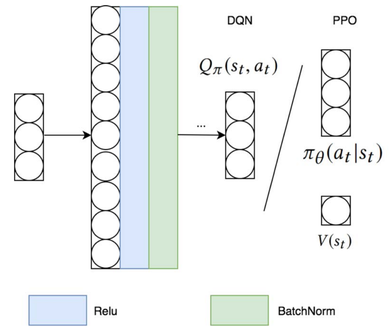
$$Q_\pi(s_t, a_t) = r_t + \gamma * Q_\pi(s_{t+1}, \pi(s_{t+1}))$$

Then, updating Q function becomes a problem of minimizing temporal difference error:

$$\sigma = Q_\pi(s_t, a_t) - (r_t + \gamma * max_a Q'_\pi(s_{t+1}, a))$$

Reinforcement Learning Structure

A policy π is learned by a Multilayer Perceptron (MLP) for choosing the next action to maximize reward.



DQL+Pop-art

Pop-art is a method that adaptively normalizes the state-action value function used in the learning updates is proposed.

Algorithm 1 SGD on squared loss with Pop-Art

For a given differentiable function h_θ, initialize $\boldsymbol{\theta}$.
Initialize $\mathbf{W} = I$, $\boldsymbol{b} = \mathbf{0}$, $\boldsymbol{\Sigma} = I$, and $\boldsymbol{\mu} = \mathbf{0}$.
while learning **do**
  Observe input X and target Y
  Use Y to compute new scale $\boldsymbol{\Sigma}_{\text{new}}$ and new shift $\boldsymbol{\mu}_{\text{new}}$
  $\mathbf{W} \leftarrow \boldsymbol{\Sigma}_{\text{new}}^{-1}\boldsymbol{\Sigma}\mathbf{W}$, $\quad \boldsymbol{b} \leftarrow \boldsymbol{\Sigma}_{\text{new}}^{-1}(\boldsymbol{\Sigma}\boldsymbol{b} + \boldsymbol{\mu} - \boldsymbol{\mu}_{\text{new}})$ (rescale $\mathbf{W}$ and $\boldsymbol{b}$)
  $\boldsymbol{\Sigma} \leftarrow \boldsymbol{\Sigma}_{\text{new}}$, $\quad \boldsymbol{\mu} \leftarrow \boldsymbol{\mu}_{\text{new}}$ (update scale and shift)
  $\boldsymbol{h} \leftarrow h_\theta(X)$ (store output of h_θ)
  $\boldsymbol{J} \leftarrow (\nabla_\theta h_{\theta,1}(X), \ldots, \nabla_\theta h_{\theta,m}(X))$ (compute Jacobian of h_θ)
  $\boldsymbol{\delta} \leftarrow \mathbf{W}\boldsymbol{h} + \boldsymbol{b} - \boldsymbol{\Sigma}^{-1}(Y - \boldsymbol{\mu})$ (compute normalized error)
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \boldsymbol{J}\mathbf{W}^\top\boldsymbol{\delta}$ (compute SGD update for $\boldsymbol{\theta}$)
  $\mathbf{W} \leftarrow \mathbf{W} - \alpha\boldsymbol{\delta}\boldsymbol{h}^\top$ (compute SGD update for $\mathbf{W}$)
  $\boldsymbol{b} \leftarrow \boldsymbol{b} - \alpha\boldsymbol{\delta}$ (compute SGD update for $\boldsymbol{b}$)
end while

By definition, action-state value function and state value function are:

$$Q_\pi(s_t, a_t) := E_{s_{t+1}, a_{t+1}, \ldots}[\sum_{l=0}^{\infty}(\gamma)^l r_{t+l}]$$

$$V_\pi(s_t) = E_{a_t, s_{t+1}, \ldots}[\sum_{l=0}^{\infty}(\gamma)^l r_{t+l}]$$

We empirically estimate value function by General Advantages Estimation (GAE):

$$\hat{A}_t = \delta_t + (\lambda * \gamma)\hat{A}_{t+1}$$

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

We integrate trading cost to reward by the following equation:

$$c_t(n) = C * (|n| + 0.01n^2)$$

PPO

Proximal Policy Optimization improve a surrogate objective vial stochastic gradient descent:

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta), clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t]$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}$$

We used GAE to estimate advantage functions. By adding value function regression loss and entropy bonus we have the loss function:

$$L_t(\theta) = \hat{E}_t\left[L_t^{\text{CLIP}}(\theta) - c_1 L_t^{\text{VF}}(\theta) + c_2 S[\pi_\theta](s_t)\right]$$

Where value function loss is given by $\left(V_\theta(s_t) - V_t^{\text{targ}}\right)^2$

Conclusion & Discussion

- **Training time and Convergence**
  - DQL and PPO with reward clipping will have crashed reward if training takes too long. DQL with Pop-art can fix this issue. In general, PPO has the fastest convergence speed than all other methods.
- **Accumulative Reward**
  - All DRL agents have a similar policy to baseline delta hedging.
- **Total Cost and Volatility**
  - Non-cost case (cost multiplier = 0)
    - All DRL agents find more optimal strategy as the average realized vol are much lower compared to baseline delta, but slightly large than zero as financially, given discrete time trading, DRL agents tend to be off a bit in between hedging time.
  - High-cost case (cost multiplier =5)
    - Delta trades too much inducing higher cost compared to all DRL agents.
    - All DRL agents realize much lower average cost while maintaining the hedge, showing their capability balancing between trading error and costs.
    - Overall PPO achieves better performance in terms of its lower average cost at 54.87 and standard deviation at 12.70, compared to both DQL and DQL with Pop-art, at the sacrifice of slightly higher volatility of total P&L, representing its more cost-conscious decision at trade-off between costs and trading errors.
- **P&L**
  - The mean of Delta Policy's P&L is significantly smaller than zero in both non-cost and high-cost cases
  - All DRL agents outperform Delta as their t-statistic of P&L are much more often close to zero and insignificant
  - DQL with Pop-art performs slightly better in high-cost case compared to other DRL agents, as it achieves significantly positive t-statistics of total P&L
- **Policy**
  - All agents trade less when cost is implemented and the number of random actions (individual dots deviating from the piecewise segments) decreases.
  - PPO trades more conservatively than both DQN agents. There is an apparent effect of Pop-art on regular DQL: less random actions are made, and the decisions are closer to the benchmark decisions (dashed lines, derived from delta hedging.).

Results

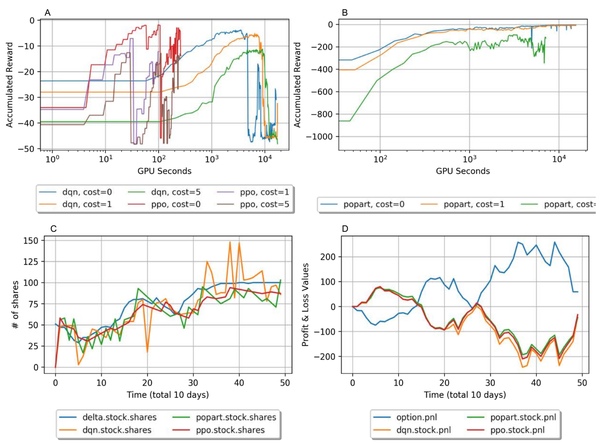


Figure 1. A) Reward convergence of DQN and PPO. **B)** Reward convergence of DQN with Popart. DQN and PPO with reward clipping will have crashed reward if training takes too long. DQN with Popart can fix this issue. In general, PPO has the fastest convergence speed than all other methods.
C) Out-of-Sample simulation in position of stocks given by DRL methods and baseline delta policy. **D)** Profit & loss simulation given by DRL methods and actual option profit and loss. The figure suggests all DRL agents have a similar policy to baseline delta hedging.

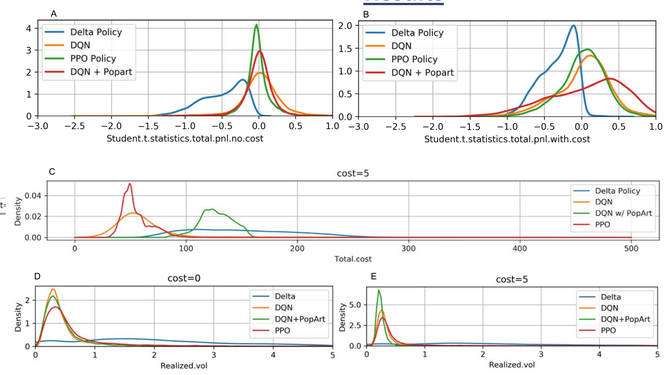


Figure 2. A) B) Kernel Density Estimates of the t-Statistic of Total P&L without cost (A) and with cost (B) for Delta, DQL, DQL with Pop-art and PPO from N=7,000 Out-of-Sample simulations
C) Kernel Density Estimates for Total Cost of Total P&L without cost for Delta, DQL, DQL with Pop-art and PPO from N=7,000 Out-of-Sample simulations
D), E) Kernel Density Estimates for Volatility of Total P&L without cost (D) and with cost (E) for Delta, DQL, DQL with Pop-art and PPO from N=7,000 Out-of-Sample simulations

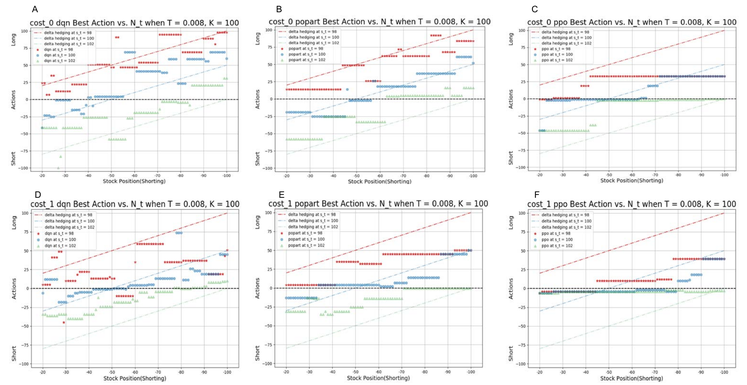


Figure 3. Policy Plots under Strike Price(K) = 100, Time to Maturity(T) = 0.008. x-axis denotes stock position (shorting 20 to 100 stocks) and y-axis denotes actions determined by agent.
A) DQN policy plot with no cost, **B)** DQN with PopArt policy plot with no cost, **C)** PPO policy plot with no cost,
D) DQN policy plot with no cost, **E)** DQN with PopArt policy plot with no cost, **F)** PPO policy plot with no cost.
The figures show that the PPO agent will make the most conservative actions, and the DQN with PopArt will make more stabilized decisions than DQN.
The implementation of cost acts as a stabilizer where all models decide to trade less given the same scenario and there are less random actions.

Acknowledgements

It is our pleasure to have our wonderful advisor Petter N.Kolm leading and supporting us throughout the project.