Deep Q Learning for Optimal Liquidation

Patrick Durojaiyes
Department of Computer Science
University College London
patrick.durojaiye.23@ucl.ac.uk

March 25, 2024

Abstract

This paper discusses the use of Reinforcement Learning (RL), specifically Deep Q Learning (DQL) in regards to optimal liquidation, of a large inventory within a given trading day for the digital assets market. This model aim's to compete with the Volume Weighted Average Price (VWAP) execution algorithm. For the sake of our analysis we apply the model to the cryptocurrency token $BLUR for the BLUR/USDT market

1 Introduction

Given the complex and stochastic nature of the financial markets, undertaking the process of liquidating a large inventory of assets poses a serious dilemma. The forever changing process of the markets causes liquidating large inventory to face the following challenges; time and execution risk, as well as liquidity. These dilemmas are ones that portfolio managers, market makers and various other market participants often face. There has been well established solutions to this problem such as the time weighted average price (TWAP), volume weighted average price (VWAP) and the Almergen and Chriss model [1]. However these models have limitations, with the Almergen and Chriss model not being able to adjust the varying market conditions over time as it is based on several assumptions and the VWAP also not being able to adapt to the dynamic market conditions.

Over the years with the development of machine learning there has been an increased focus on exploring machine learning specially RL in the theme of order execution. Brian, Franco and Sebastian [5] were the first to propose the concept of DQL to order execution. They use a double deep q network and experience replay to estimate the optimal actions to execute orders and showed it outperformed TWAP strategies. This research was further built upon combining deep learning and hierarchical RL joint architecture for VWAP

strategy optimization [3], results of this study highlighted reduced execution costs upon utilising RL instead of a vanilla VWAP strategy.

Previous studies have shown the advantage DQL has over traditional methods for optimal liquidation. We propose a DQL model that learns the most optimal actions to liquidate a large inventory within 24 hours with the aim to outperform the VWAP. This model is able to adapt to various market dynamics and conditions.

2 Background

2.1 VWAP

VWAP was created by the trading firm Abel Noser in 1984 [2] and has since been used as an industry standard to determine fair price of an asset and if the executed price was executed well. VWAP is formulated via the following;

$$VWAP = \frac{\Sigma(Price \cdot Volume)}{\Sigma Volume} \tag{1}$$

The price used in the VWAP is the average of the Low, High and Close at each given time step.

2.2 Exponential Moving Average

Exponential moving average (EMA) is a moving average of a time series, while giving more weight to recent occurrences. When price time series it can be utilised to determine if a market is trending. If at time t price is above the EMA this indicates that the market is trending upwards and vice-versa if price is below EMA.

$$EMA_t = (Price \cdot \alpha) + (EMA_{t-1} \cdot (1 - \alpha)) \tag{2}$$

2.3 Reinforcement Learning & Q Learning

Reinforcement learning is a field of machine learning where an agent takes a set of actions within a given environment. The agent is given a policy which it utilises to guide itself to select an action given its current state in the environment. If the agent is successfully taking good actions it is rewarded, with the aim of the agent to maximises its reward over the course of time.

In Q learning a model decides on an action based on which action provides the highest Q-value. Q-value is a measure of quality of an action in a particular state undergoing some policy π. The Q learning model does not learn a policy rather it learns which action provides the best value uses this to make decisions.

$$Q(s,a) = E_\pi[R_{t+1} + \gamma V(S_{t+1})|S_t = s, A_t = a] \tag{3}$$

Equation 3 is the equation used to measure a Q-value and is known as the Bellman equation. Where R are rewards, V is the value of a given state, a being an action and γ being the discount factor. Q learning is an iterative process that is initiated with an initial estimate for Q and then relies on the Bellman equation to update a Q-table which is a matrix of q values for given states and actions.

$$Q_{new}(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma maxQ^{'}(s',a') - Q(s,a)] \tag{4}$$

Equation 4 is Bellman's equation used to update Q values in the Q table for each state and action. Upon the Q-table being populated the agent uses the Q-table as a look up table to select the action with the highest q value.

2.4 Deep Q Learning

DQL is a form of RL that was introduced by the Google deep mind in 2015 [4]. DQL builds upon Q learning, however it incorporates a neural network instead of a Q-table. This neural network is used to approximate the Q-function. DQL has an advantage over Q learning as Q learning struggles to perform for complex environments as well as large state sizes.

In DQL a neural network takes a state as an input and maps such input to a Q-value for each possible action an agent can take. DQL models compose of two neural networks, a primary network used for training to estimate the Q-value function and another used for generating target Q-values for the primary Q-network to use for its loss function in training. Two neural networks are used in DQL to stabilise the Q-function updates and aids the learning of the model.

$$L(\theta) = \Sigma_i(Q^*(s_i,a_i) - Q(s_i,a_i,\theta))^2 \tag{5}$$

Equation 5 is the loss function of for the DQL network, with θ being the parameters of the neural network used to estimate Q-values. The networks weights are updated based on the loss computed between the predicted Q values and the target Q values. Experience replay is a technique utilised by DQL networks, this technique removes the correlation between states. It does this by storing experiences in a replay buffer and randomly sampling mini-batches of experiences from such buffer. This technique allows for better stability upon training a model.

3 Methodology

3.1 Feature Engineering

We take into account an agent who wants to sell X amount of $BLUR tokens within a 24 hour period, this agent has two actions it can take which is to

hold or to sell a portion of tokens being held in inventory. For our analysis we use 5 minute interval Open, High, Low, Close, Volume (OHLCV) data of the BLUR/USDT market on Binance from the 30th of November 2023 to the 29th of December 2023. This data was sourced from Binance via CCXT. Upon downloading our data and analysing it we had discovered 16 duplicate rows which were groups of 2 duplicates each. To clean this we dropped the duplicates and kept the first series we saw out of each batch of duplicates. Limit order book data was not included as a feature as our analysis assumes zero market impact when executing orders.

Using the formula to calculate VWAP in equation 1 we calculate the VWAP for each time step of our dataset. However we reset the VWAP at the first 5 minute interval of each day as VWAP is an intra day indicator. We also incorporate the standard deviation (std) of the VWAP at each time step of the day, this is done as we use $\pm\frac{1}{4}$ of the std in our reward mechanism.

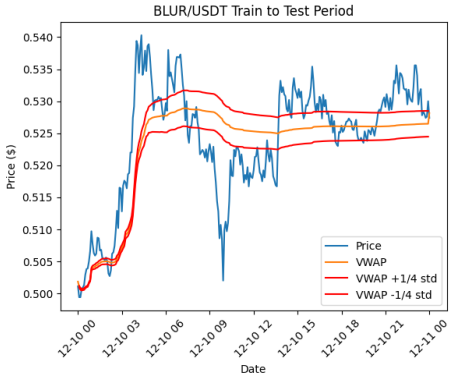

Figure 1: BLUR/USDT price series

We also incorporate a EMA as a feature into our model. We want our agent to be able to liquidate assets during varying market conditions. The EMA feature will allow our agent to determine if the market is trending upwards or downwards and liquidate depending on how the market is trending. As our model is applied to crypto currencies which often face trending market conditions this feature is a key feature when considering optimal liquidation.

3.2 Neural Network Architecture

Our neural network architecture incorporates an input layer consisting of 9 inputs representing each element of our state. We include 2 hidden layers in our neural network with both compromising of 20 neurons each. Each hidden layer was activated by using a Rectified Linear Unit (ReLU) activation function. ReLU is used to introduce non linearity into our neural network, this aids in the

network learning complex relationships. The output layer has 2 neurons each for representing the actions to hold or sell. The linear activation functions estimate the Q-values for each action in a given state. Our model is compiled with an Adam optimiser and mean square error for its loss function. An Adam optimiser was chosen as it dynamically adjusts the learning rate of the model, allowing the avoidance of gradients blowing up while training.

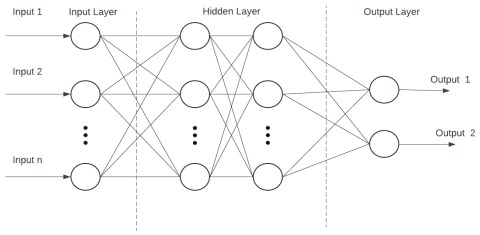

Figure 2: Neural Network Design

3.3 State

States in our model allow for our agent to be aware of the environment it currently is operating in. Our state space at each time step includes the close price, volume, VWAP, EMA, current $USDT holdings, inventory and followed by binary indicators if close price is above VWAP and if the market is trending. The market related states were chosen so our agent is aware of market states when making decisions. Inventory and current $USDT holdings make our agent aware of how much inventory we have left to sell along with how much inventory we have liquidated.

3.4 Actions

Our agent can take on at most 2 actions at each time step. They are to either sell or hold. When the agent is first initiated it is given a base amount of $6944.\dot{4}$ to sell. However depending on the state the agent will change the amount it aim's to sell.

Figure 3 illustrates how our agent determines how much inventory to sell based off our state. Our agent sells aggressively if the price at the given state is above the VWAP and the market is trending. This would be the most optimal liquidation hence we sell 60% more than the base amount.

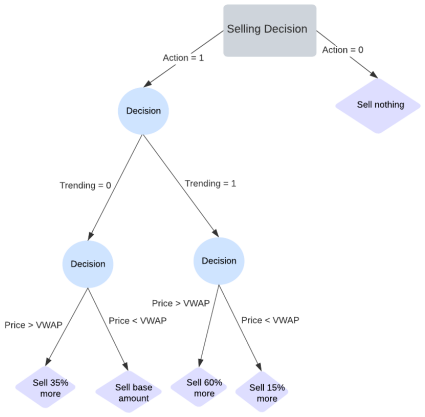

Figure 3: Selling decisions based on state

3.5 Rewards

Given that our agent is trying to maximise the its reward based on the derived policy from Q-values, the reward mechanism design is crucial. Our agent is given both positive and negative rewards based on actions it takes and the state that it is in.

$$R_{\text{hold}}(s,a) = \begin{cases} +(s_0 - s_2) \cdot \text{sell amount} & \text{if } a = 0, s_6 = 1, s_5 = 1, \text{inventory\_ratio} < 0.4 \\ +(s_0 - s_2) \cdot \text{sell amount}/2 & \text{if } a = 0, s_6 = 1, s_5 = 1, \text{inventory\_ratio} \geq 0.4 \\ -(s_0 - s_2) \cdot \text{sell amount} & \text{if } a = 0, s_6 = 0 \text{ and } s_5 = 1 \\ -(s_2 - s_0) \cdot \text{sell amount} & \text{if } a = 0, s_6 = 0, s_5 = 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$R_{sell}(s,a) = \begin{cases} +(s_0 - (s_2 - s_3)) \cdot \text{sell amount} & \text{if } a = 1,\ s_0 > s_2 + s_3 \\ -((s_2 + s_3) - s_0) \cdot \text{sell amount} & \text{if } a = 1,\ s_0 < s_2 - s_3 \\ +((s_2 + s_3) - (s_2 - s_3)) \cdot \text{sell amount} & \text{if } a = 1,\ (s_2 + s_3) \leq s_0 \leq (s_2 - s_3) \end{cases} \tag{7}$$

Where s_0 = close price, s_2 = VWAP, s_3 = VWAP $\frac{1}{4}$ std, s_5 = Indicator if price is above VWAP, s_6 = Indicator if market is trending up or down

The above is base of our rewards mechanism. We give our agent small rewards for holding to not incentivise the agent to hold onto the inventory for too long over time due to our task being optimal liquidation. Negative rewards are

assigned to the agent for holding onto assets if market is not trending and price is above or below the VWAP. We award our agent positive rewards if it liquidates assets above the VWAP or between $\pm\frac{1}{4}$ std away of the VWAP. We also assign a positive reward for the reduction of our inventory to incentivise liquidation of the inventory held. As our objective is to hold 0 inventory at the end of the day we give a positive reward for having 0 inventory before the end of the day or at the end of the day. While we providing a strong penalty in reward if we still have inventory at the end of the day.

3.6 Epsilon Greedy Algorithm

Deep Q Learning models often face a problem of over exploration and exploitation. Where the agent explores to many possible actions or the agent continuously exploits its actions that its taken that gave high rewards without exploring. When training our model opt to aim for a balance between exploration and exploitation. To achieve this we avail of the epsilon greedy algorithm.

Algorithm 1 Epsilon Greedy Algorithm

```
for each time step do
    Generate a random number between 0 and 1
    if number < ϵ then
        Pick a random action
    else
        Pick action that yields the max Q-value
    end if
end for=0
```

ϵ is a probability that an agent chooses a random action. Its value is constantly decaying. We opt for an exponential decay of ϵ. Where ϵ is decaying by 0.822 each time step. Eventually as ϵ decays to its terminal value of 0.01 it will focus less on exploring and utilise its experiences to produce an action that yields the highest Q-value.

4 Evaluation Results

4.1 Deep Q Learning Results

Our model was trained from 1st December 2023 to 7th December 2023, giving an initial inventory to liquidate of 2,000,000 $BLUR tokens. It was trained on one Nvidia H100 SXM GPU with a batch size of 16 and learning rate of 0.05 for 15 episodes.

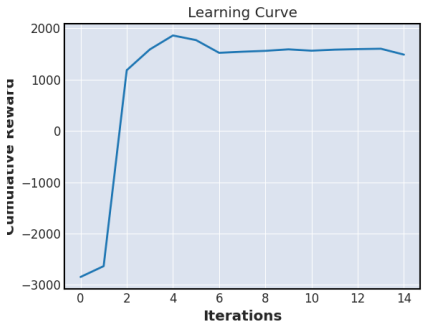

Figure 4: Learning curve

We found after the 7th iteration our model stabilises as the learning curve plateaus. We can confidently say our model has learnt the best actions to take to efficiently liquidate its inventory.

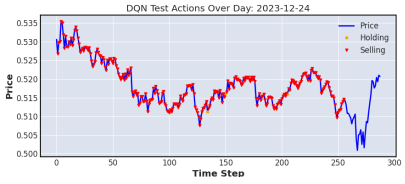

Figure 5: Actions during Liquidation

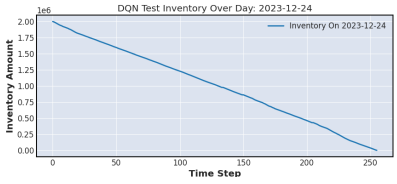

Figure 6: Inventory reduction

As price series tend to have higher correlation when analysed at shorter intervals we used testing dates 2 weeks away from the last day we trained on to reduce correlation between train and test data. We test our model from the 20th December 2023 to 27th December 2023 with an initial inventory equal to that of our training. The remainder inventory at the end of the day for each test trading day was 0, indicating that our model was working as intended. Our model was also able to identify optimal times to hold instead of selling.

4.2 Baseline Model

To evaluate our DQL models performance we compare it to a baseline model. Our baseline model is a simplistic VWAP algorithmic strategy.

This VWAP algorithim is highly simplistic. It does not incorporate any market data other than price and volume and also ignores market impact of sells. For the purposes of our evaluation is tested on the same out of sample data as our DQL model for.

Algorithm 2 Simplistic VWAP Algorithim

```
for each time step of the tading day do
  if Price > VWAP then
    Sell 1.1x more quantity
  else
    Sell fixed quantity
  end if
end for=0
```

4.3 Results

As we are dealing with optimal liquidation, the VWAP of execution as well as execution efficiency are two key metrics in regards to the evaluation between our DQL and baseline model. We compute VWAP of execution and execution efficiency in the following manner. For both models we take the time series data of tokens sold over time as well as price sold. This is used to create a VWAP of our executions and compare to the VWAP of that day, which is our benchmark.

$$Execution\ Efficiency = VWAP_{model} - VWAP_{benchmark} \tag{8}$$

We obtained the Execution efficiency for various liquidation events of the DQL and Baseline VWAP for statistical analysis of our models.

Method	Statistic	Value
DQL	Mean	0.0002
	Median	-0.0004
	Std	0.0016
Baseline VWAP	Mean	-0.0007
	Median	-0.0008
	Std	0.0017

Table 1: Statistics of DQL model vs Baseline VWAP.

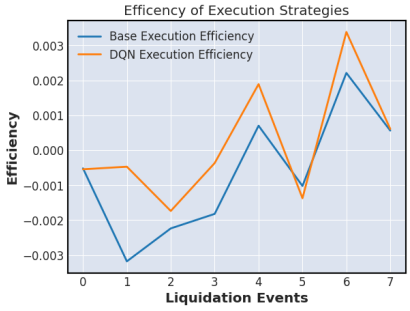

Figure 7: Execution efficiencies over liquidation events

Table 1 and Figure 7 show that on average the DQL execution offers a better liquidation as supposed to the baseline VWAP. However to conclude this with strong evidence we use more robust statistical measures to inform if the DQL offers a better liquidation compared to our baseline model. We will be undergoing a hypothesis test.

We first test for gaussianity in the efficiency over liquidation events for both our DQL model and baseline model. Shapiro Wilk test is a statistical test to determine if sample of data is derived from a gaussian distribution. The null hypothesis (H_0) of the test is the sample data follows a gaussian distribution, with the alternative hypothesis (H_a) being the sample data does not follow a gaussian distribution. Upon determining if both our models follow a gaussian distribution or not we perform a another hypothesis test to analyse measure how statistically different the efficiency of both models are.

If the sample data of efficiency of both DQL and baseline VWAP follow a gaussian distribution we use conduct a student t-test however if one or both do not follow a gaussian distribution we conduct a Mann–Whitney U test. These test both have a H_0 that is there is no significant difference in efficiency and a H_a that there is a significant difference in efficiency.

The distribution of our execution efficiency for both models follow a gaussian distribution. A t-test using both distributions returns a p-value of 0.3541 causing us to not having enough evidence to reject H_0. This means that while on average DQL has a better execution than the baseline VWAP, statistically the difference in the gain of execution efficiency is insignificant.

5 Conclusion

This paper we discussed optimal liquidation in the context of DQL. The results of the study show that the utilising DQL for optimal liquidation does outperform the traditional benchmark of the VWAP on average. However the difference in efficiency of liquidation between a complex model such as the DQL and a simplistic VWAP execution algorithm is minuscule. Given the complexity and the computational cost needed to run a DQL model if one does not have the computational power to run a DQL they should utilise a simplistic VWAP execution. However as our analysis was based on one market it would be beneficial to access both models across other markets to make a more informed decission on which model offers better efficiency. The results of this study do show promising results in the efficiency of using DQL for liquidation of assets. Further research could build upon this study to achieve greater effiency in liquidation by predicting the VWAP using models such as a LSTM and feeding this prediction into our DQL network as an input. This would allow our agent to make more informed decisions based on future expected market conditions.

References

[1] Robert Almgren and Neil A Chriss. Optimal liquidation. https://dx.doi.org/10.2139/ssrn.53501, 1997.

[2] Capital.com. What is vwap? https://capital.com/vwap-definition, 2024.

[3] Xiaodong Li, Pangjing Wu, Chenxin Zou, and Qing Li. Hierarchical deep reinforcement learning for vwap strategy optimization, 2022.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[5] Brian Ning, Franco Ho Ting Lin, and Sebastian Jaimungal. Double deep q-learning for optimal execution, 2020.