Algorithmic Trading Using Reinforcement Learning

Nivedita Ganesh

New York University

nive@nyu.edu

February 13, 2020

Structure of Presentation

1. Motivation
2. Intro to deep Q learning
3. Problem Formulation
4. Results

Why use AI for algorithmic trading?

- Algorithmic trading comprises of 70% trading in the US markets.
- Vast majority of algorthmic trading strategies comprises of relative value strategies which are largely based on *convergence to mean*.
- Algorithmic trading has two main components:
  - **Policy**: actions chosen by traders
  - **Mechanism**: implemented by machines

What is Deep Q Learning?

- In the Q-Learning Algorithm, the Q Function is used to approximate the reward based on a state. We call it Q(s,a), where Q is a function which calculates the expected future value from state s and action a.
- Similarly in Deep Q Network algorithm, we use a neural network to approximate the reward based on the state.

Deep Q Learning

The loss is just a value that indicates how far our prediction is from the actual target. For example, the prediction of the model could indicate that it sees more value in buy when in fact it can gain more reward by doing nothing. We want to decrease this gap between the prediction and the target (loss). We will define our loss function as follows:

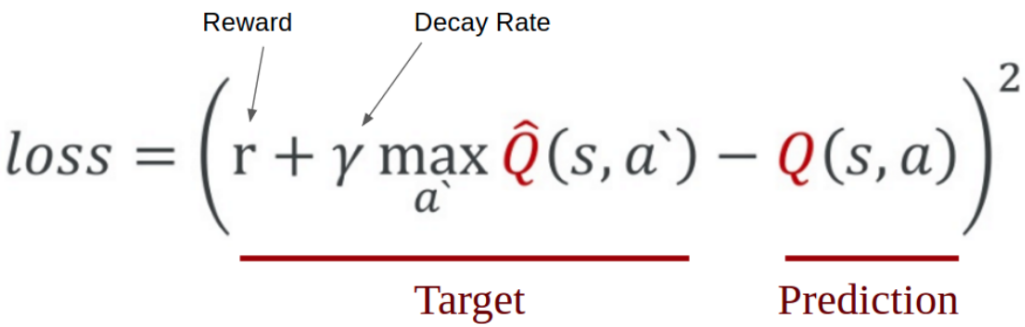

Figure: Q learning equation

Pseudocode

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Figure: Pseudocode for Deep Q Learning

Problem Statement

- To find a trading strategy via deep-q-networks
- We propose a MDP suitable for the trading task and solve it using deep q-network.
- Interested in finding the short term trends - local peaks and troughs

Dataset

- Cryptocurrency: Bitcoin and Ethereum
- NSE Equity Instrument - ITC
- Frequency : Daily Close prices
- Time-period : Jan 2010 - Current Data

Input

- Equity Instrument
- n : n-day windows of closing prices to determine the best action.
- Transaction cost
- Available Liquidity : Maximum amount to trade in the market

Assumptions

- Our impact on the market is negligible
- The price at which an order is executed is the last close price
- No short selling
- Liquidity constraint : Maximum number of unsold stocks in the portfolio is limited to 50
- Transactions cost is fixed at 0.5%

MDP Formulation

- State-space $\in [0,1]^n$: Sigmoid activation of n consecutive pairwise differences of closing prices
- Action-Space $\in \{buy, sell, do - nothing\}$
- Reward
  - Sell : Profit earned(Current price - earliest bought price)
  - Buy/Do nothing : 0

Hyper-Parameters

- **episodes** - Number of iterations we want the agent to play/train.
- **gamma** - Discount rate, to calculate the future discounted reward.
- **epsilon** - Exploration rate, this is the probability with which an agent randomly decides its action rather than prediction.
- **epsilon_decay** - The decay of the exploration rate.
- **epsilon_min** - Minimum amount the agent must explore.
- **learning rate** - Gradient descent learning rate of the neural network.

Neural Network - Model Architecture

Layer	Input Dimensions	Output Dimensions	Activation
1	n-day window	64	relu
2	64	32	relu
3	32	8	relu
4	8	3	linear

Training

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Figure: Pseudocode for Deep Q Learning

Testing

- For each trading day:
  - Compute the state using previous n-day window
  - Pass the state as input to the neural network and get predicted rewards of different actions
  - If the optimal action is sell and we have a corresponding stock in our portfolio, execute the action and collect reward
  - If the optimal action is buy and the total number of stocks in portfolio is less than 50, execute the action and add to portfolio
  - Otherwise, do nothing

Hyper-parameter values

Hyper-parameter	**Value**
episodes	1000
gamma	0.95
epsilon	1
epsilon_decay	0.995
epsilon_min	0.01
learning_rate	0.001

Results

Instrument	Test Data	Total Profit	Trades	Liquidity
BTC	Jan '17 - curr	$38,264.77	463	$270,858
ETH	Jan '17 - curr	$8,338	183	$ 20,098
ITC	Jan '15 -curr	INR 4,939	197	INR 11,558

Table: Result on unseen data

Results - Graphs

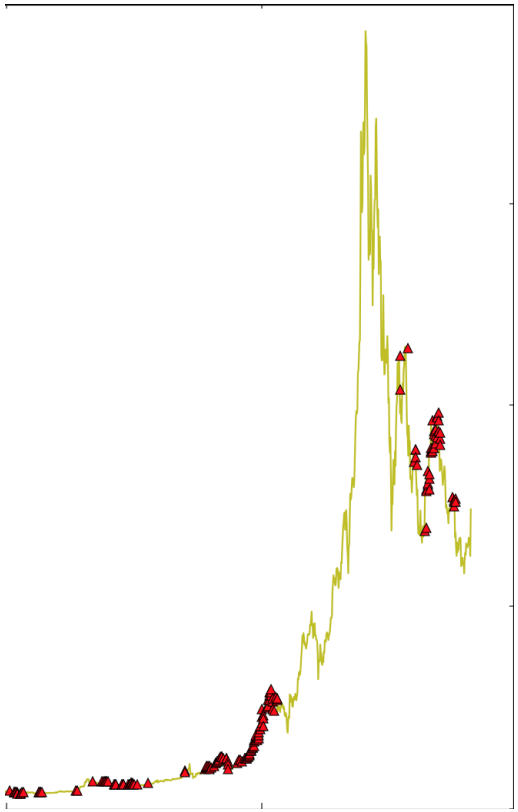

Figure: Bitcoin - Price vs Time graph marked with agent's buy action

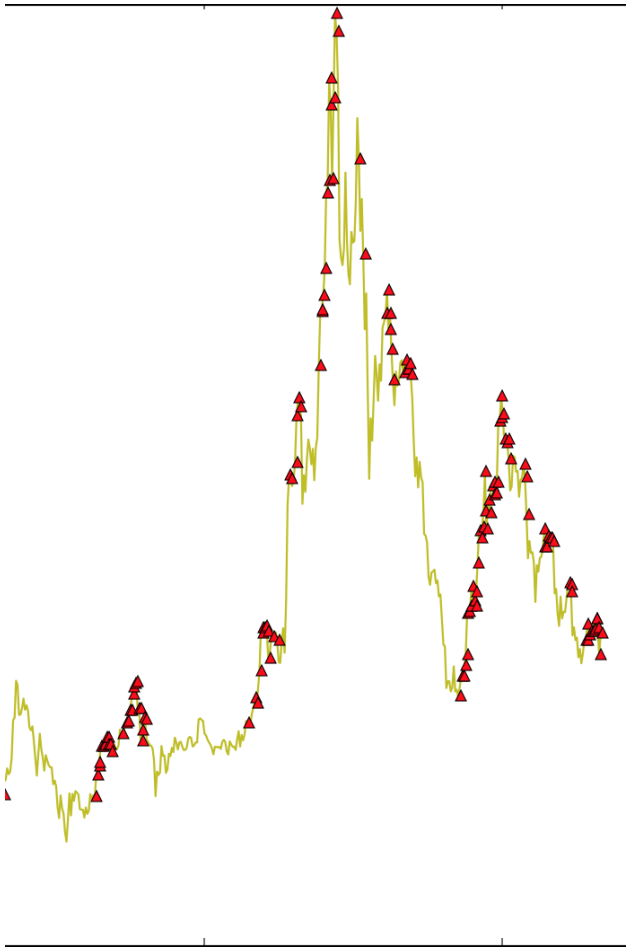

Figure: Ethereum - Price vs Time graph marked with agent's buy action

Results - Graphs

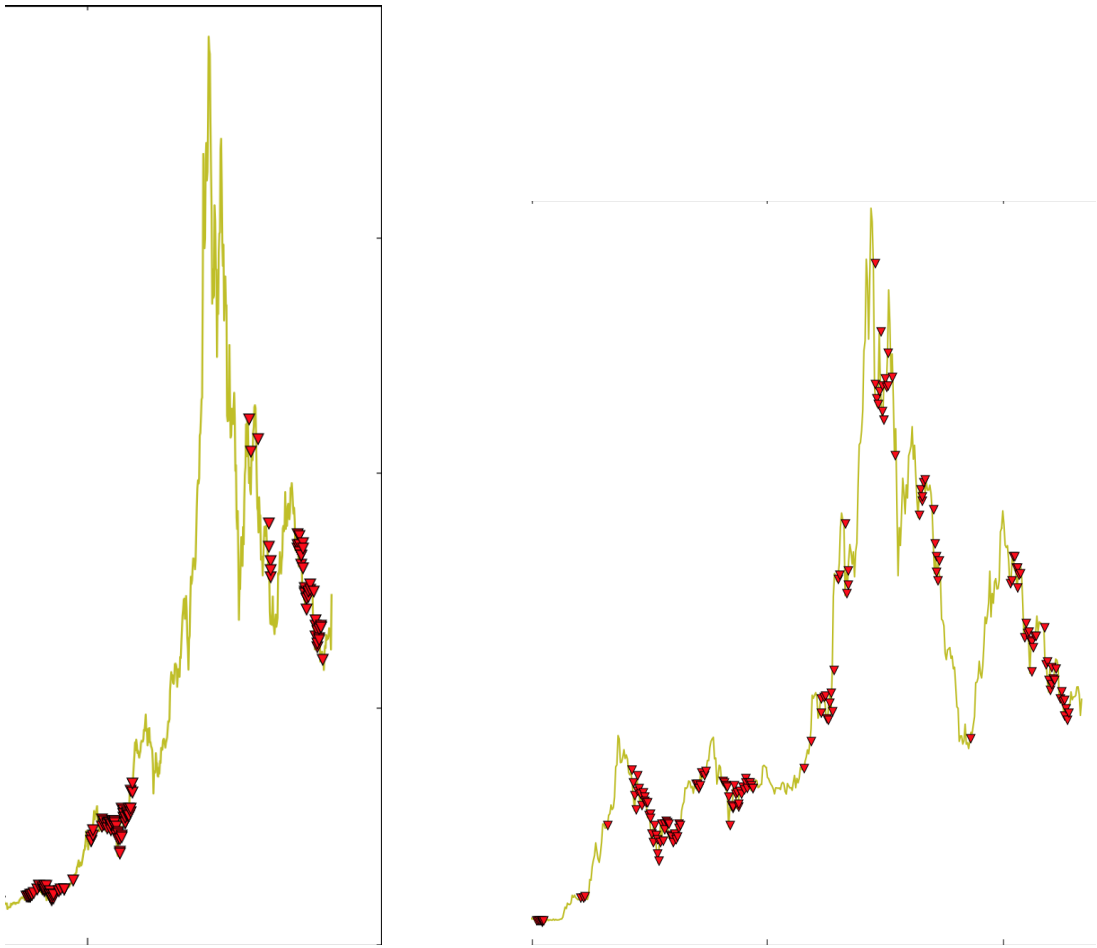

Figure: Price vs Time graph marked with agent's sell action

Results - Graphs

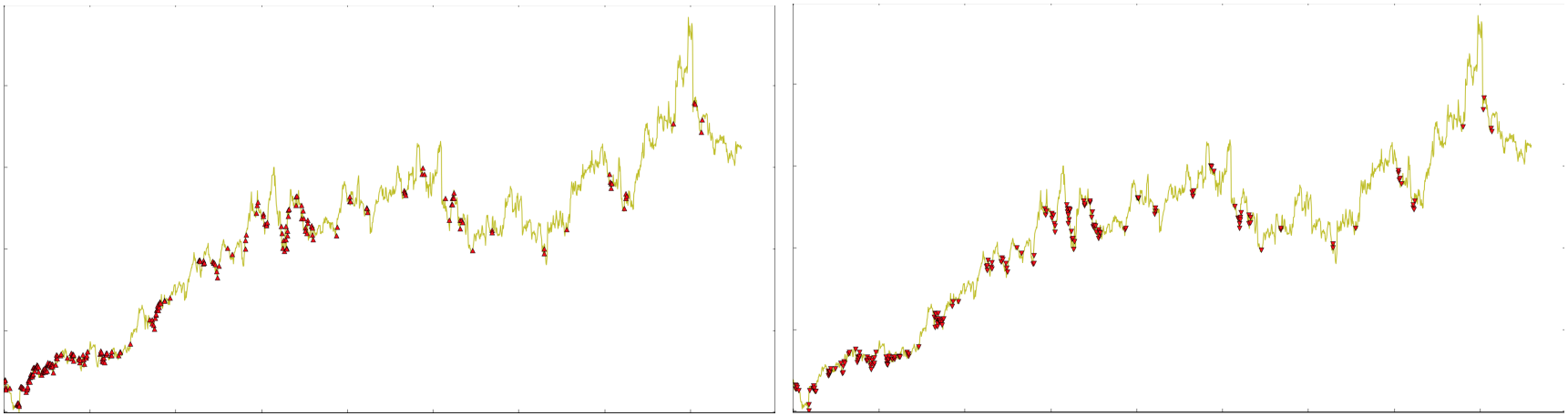

Figure: ITC(NSE) - Price vs Time graph marked with agent's action

The End