Automated Trading using Q Learning

UDACITY

Machine Learning Engineer Nanodegree

Capstone Project

Arijit Santra

Contents

Definition

Project Overview

Automated trading algorithms have made their foray into the financial world. Over the years, the world of trading has evolved to leverage technology to create a differentiated advantage. In the recent past, machine learning has emerged as powerful weapon for trading firms to generate alpha and reduce costs. These tools have expanded across the domains of trading strategy, trade execution, settlement, compliance and operations.

So I have created an automated trading agent which made use of Q learning to predict trades and outperformed the Indian Nifty index. The trading agent used historical data of Nifty Index, VIX and technical indicators to come up with the best possible action for a given state (combination of VIX and technical indicators). The in-sample data was taken from 1st January 2017 to 31st December 2017. From this in-sample data, the trading agent generated the Q-table which is the combination of various state-action pairs. Based on this Q table, the trading agent tried to predict trades for the out-of-sample period from 1st January 2018 to 30th June 2018 and outperformed the Nifty index in most of the cases.

Problem Statement

The goal is to create an automated trading agent which will use Q learning (Reinforcement Learning) to predict trades for the Indian Nifty index, based on historical data and technical indicators (moving average, RSI, MACD).

In Q learning, the agent tries to learn the optimal policy based on its interaction with the historical states. The agent generates the Q table which is the combination of state-action pairs using training data and uses that Q table to identify policies for states in the test data.

Here, the states are the combinations of discretized 5 day moving average, 10 day moving average, 20 day moving average, RSI, MACD and VIX. The trading agent will try to come up with the optimal action i.e. either one of 'buy', 'sell', 'hold' for each state, which will maximize the reward (risk adjusted return in this case).

Metrics

The Nifty returns for a particular period of time will be used as the benchmark for the returns generated by the trading agent.

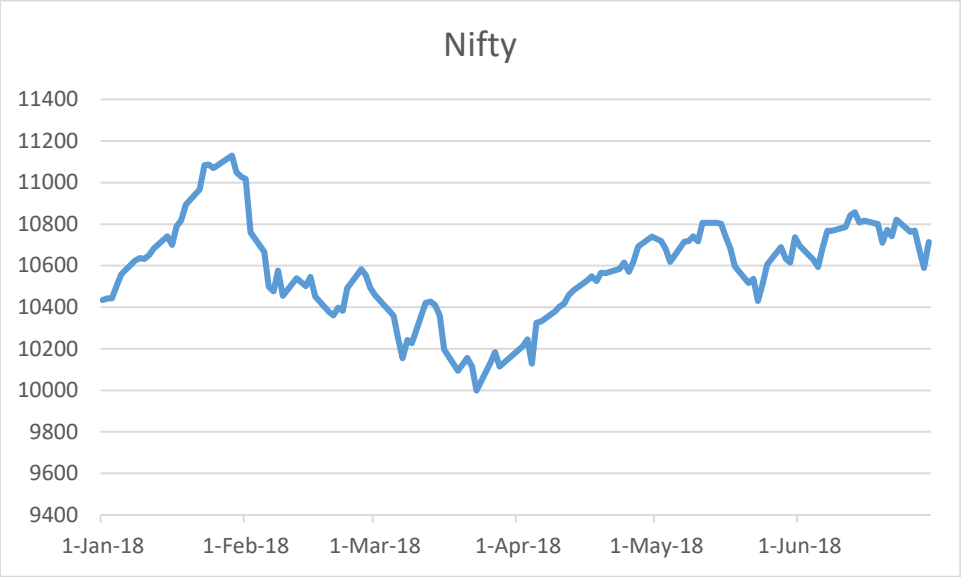
The performance of the trading agent for each trade will be measured by the Sharpe ratio. Sharpe ratio measures the risk-adjusted return of a portfolio using the following formula:

Sharpe ratio = (return – risk free return)/volatility. So the reward in the Q learning algorithm will be equal to the Sharpe ratio.

Sharpe ratio divides the return in excess of the risk free return by the volatility to prevent any over bias in the high returns that may have been achieved during periods of high volatility. So it rewards high returns, but penalizes high volatility as well. It tells us how much reward we are getting for each unit of risk we are taking. So it's a good measure for the performance of the trading agent, because the ultimate aim of the agent is to generate good returns maintaining a low volatility.

Analysis

Data Exploration



Out-of-Sample Data for Nifty Index from 1st January 2018 to 30th June 2018
Data source: NSE India

Total Nifty returns for this 6-month period= 2.939% which is lower than the risk free returns of 6.90/2=3.95%

It reaches a high of 11130.4 and a low of 9998.05 during this period.
Total standard deviation = 0.022 = 22% which is higher than the average VIX value for this period which is around 14%.

So this period exhibits low returns and higher volatility.

The Q learning algorithm will attempt to outperform this curve as well as the risk free returns. The main aim of the algorithm is to generate good returns during periods of low returns and high volatility.

Algorithms and Techniques

Q-learning is a reinforcement learning technique used in machine learning. The goal of Q-Learning is to learn a policy, which tells an agent what action to take under what circumstances. It does not require a model of the environment and can handle problems with stochastic transitions and rewards, without requiring adaptations.

The algorithm, therefore, has a function that calculates the quality Q of a state-action combination.

Before learning begins, Q is initialized to a possibly arbitrary fixed value (chosen by the programmer). Then, at each time t, the agent selects an action a_t, observes a reward r_t, enters a new state s_{t+1} (that may depend on both the previous state s_t and the selected action), and Q is updated. The core of the algorithm is a simple value iteration update, using the weighted average of the old value and the new information:

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

where r_t is the reward observed for the current state s_t and α is the learning rate

There are two important flags in a Q learning algorithm: epsilon and alpha.

Epsilon: It is the random exploration factor. It determines the extent to which random discoveries will be made. This reduces gradually with the number of trials. So the agent will make more discoveries at the start and less discoveries towards the end. A value of 0 indicates it will not make any random discoveries and will always use previous learnings, while a value of 1 indicates it will always make new discoveries.

Alpha: It is the Learning factor. It determines the extent to which previous learnings will be incorporated with the new learnings. A value of 0 indicates that it will not learn anything new, while a value of 1 indicates it will forget the previous learnings and will always learn new things.

Benchmark

The return generated by the trading agent will be compared with the Nifty return for the same time period. The risk free rate for a 1-year bond would be considered 6.90%.

Methodology

Data Preprocessing

I have used 5 technical indicators (5 day moving average, 10 day moving average, 20 day moving average, RSI and MACD) as well as VIX as the states for the Q learning agent.

N-day moving average is calculated as the average of the Nifty values for the past n days.

Relative Strength Index (RSI) is a momentum oscillator that measures the speed and change of price movements. A very high RSI value indicates that it is overbought and a very low RSI value indicates that it is oversold.

VIX is the volatility index of Indian Nifty index.

Moving average convergence divergence (MACD) is a trend-following momentum indicator that shows the relationship between two moving averages of prices. The MACD is calculated by subtracting the 26-day exponential moving average (EMA) from the 12-day EMA

Since Q learning uses discrete states, I have converted the technical indicators into discrete states based on the following conditions:

5 day moving average: If 5-day MA>current Nifty index, then it's 1, otherwise it's 0

10 day moving average: If 10-day MA>5-day MA, then it's 1, otherwise it's 0

20 day moving average: If 20-day MA>10-day MA, then it's 1, otherwise it's 0

VIX: If current VIX>average VIX for 1 year (in-sample), then it's 1, otherwise it's 0

RSI: If RSI>65, it's 2, if 50<RSI<=65, it's 1, otherwise it's 0

MACD: If MACD is positive, it's 1, otherwise it's 0

Implementation

The NIFTY 50 is the flagship index on the National Stock Exchange of India Ltd. (NSE). The Index tracks the behavior of a portfolio of blue chip companies, the largest and most liquid Indian securities. It includes 50 of the approximately 1600 companies listed on the NSE, captures approximately 65% of its float-adjusted market capitalization and is a true reflection of the Indian stock market.

Buying 1 unit of Nifty would mean buying the entire portfolio of the 50 companies based on the weightages assigned in the index.

Now, I have taken the in-sample period to be from 1st January 2017 to 31st December 2017, having a total 248 data points. Based on this in-sample period, the Q-learning agent generated the Q table consisting of state-action pairs after 1000 trials. There are 42 unique state-action pairs in the Q table based on the in-sample 'states' data. Each state can have 3 actions. So total number of state-action possibilities = 42*3 =126. So 1000 trials were more than enough to come up with the Q table.

I will only take the closing prices of Nifty to conduct the trades. Buying and selling will only happen at the closing price.

Initially, I will consider the entire asset to be invested in Nifty index. "Sell" would involve selling 'x' units of Nifty index and investing the proceeds in risk free 1-year bond. "Buy" would involve buying 'x' unit of Nifty index using funds from risk free 1-year bond. Here, I have considered two cases for 'x' (=1 or =5). So, the asset structure will be as follows:

[Funds in Nifty, Funds in Risk Free Bond, Total Funds]

Total Funds = Funds in Nifty + Funds in Risk Free Bond

A negative value for Funds in Risk Free Bond would mean that I borrowed that amount at a risk free rate. Similarly, a negative value for Funds in Nifty would mean I am short on Nifty and I have fully invested those funds in Risk Free Bond.

Initially, I have taken 50 units of Nifty index, invested in it completely, without any funds in risk free bonds. So in the training data, Nifty index on 2nd January 2017 is 8179.5. So total starting wealth = 8179.5 * 50=4,12,246.8 INR

In the testing data, Nifty index on 1st January 2018 is 10435.55. So total starting wealth = 10435.55 * 50 = 5,21,777.5 INR

I have taken the reward for each trade to be equal to the Sharpe ratio.

For any chosen action, the Sharpe ratio = (return-risk free rate)/volatility

1-day return = next day's closing asset value/today's closing asset value-1

Risk free rate for a 1-year bond = 6.9% per annum, which translates to 0.027823% per day assuming 248 days

So Sharpe ratio for any action = ((next day's closing asset value/today's closing asset value-1)-0.00027823)/VIX

So the Q learner tries to maximize the Sharpe ratio for any action that it takes for a particular state.

Using the generated Q table which is the combination of state-action pairs, the trading agent performed actions ['buy', 'sell', 'hold"] for the out-of-sample period.

Refinement

Alpha was taken to 0.75 and initial epsilon was taken to be 1.

I have assumed the tolerance level for epsilon to be 0.01.

I have set the epsilon decaying function to be equal to 0.995^trail.

0.995^trials=0.01, which implies no of trials = log(0.01)/log(0.995) = 918.73 ~ 1000 trials

So, I have set the number of trials to be 1000. So epsilon starts at 1 and decays to less than 0.01 after 1000 trials. So the agent makes new discoveries at the beginning and hardly makes any new discoveries towards the end of the 1000 trials.

Results

Model Evaluation and Validation

In the graphs below, for the test data, the green line represents the asset value generated by the Q learner from 1st January 2018 to 29th June 2018. The blue line represents the asset value if all the funds were invested in Nifty index throughout.

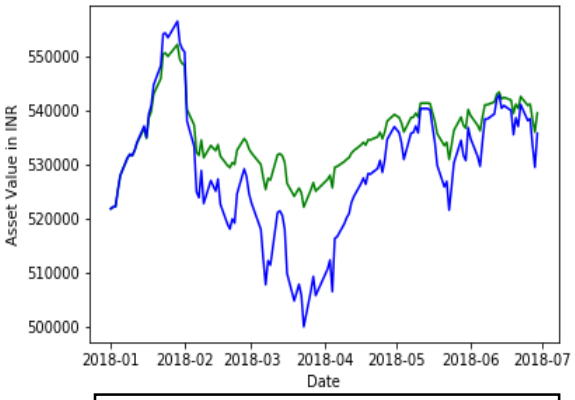
Since the Q table is slightly different for different instances of running the program, the asset value generated by the Q learner (green line) is slightly different for each case.

I have evaluated the models based on two different trade order sizes (1 unit and 5 units). So we observe that the performance of the model improves significantly if I increase the trade order size from 1 unit to 5 units. This is because limiting the trade order size to 1 unit prevents the Q learner from taking advantage of any upside potential.

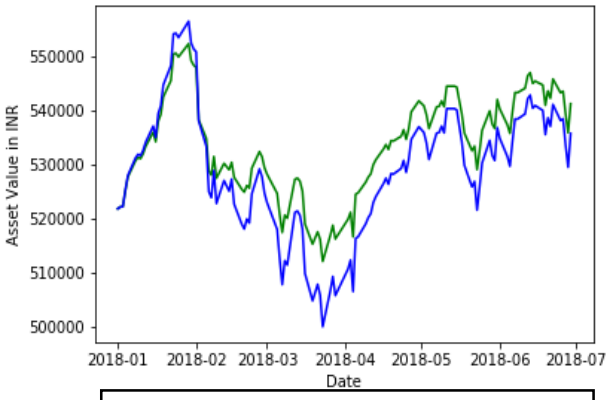
The average returns generated by the Q learner for the 6 month testing period for a trade size of 1 unit= 3-3.5% which is equal to 6-7% per annum. (Almost similar to risk free returns)

But the average returns generated by the Q learner for the same testing period for a higher trade size of 5 units = 7-8% which is equal to 14-16% per annum

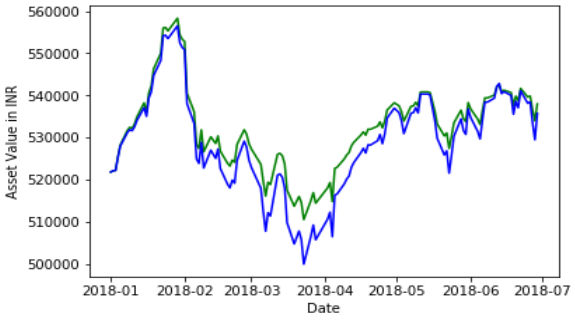
Quantity bought or sold for each trade = 1



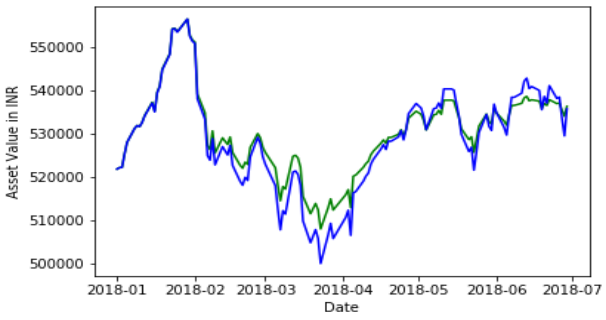
Q learner returns = 3.436%
Nifty returns = 2.939%
Q learner std deviation = 0.012
Nifty std deviation = 0.022



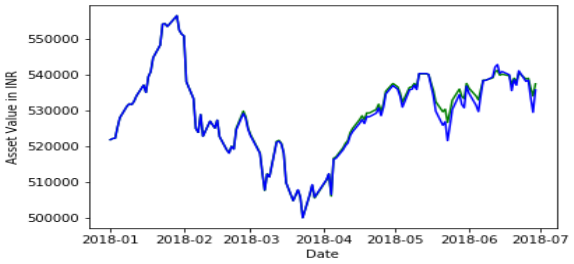
Q learner returns = 3.84%
Nifty returns = 2.939%
Q learner std deviation = 0.018
Nifty std deviation = 0.022



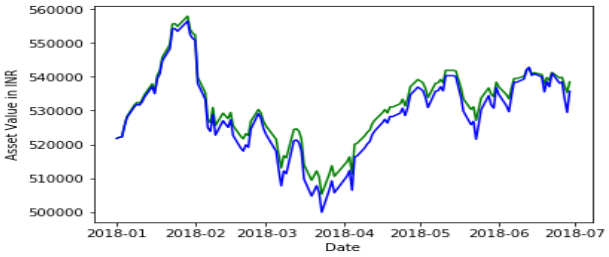
Q learner returns = 3.253%
Nifty returns = 2.939%
Q learner std deviation = 0.018
Nifty std deviation = 0.022



Q learner returns = 2.924%
Nifty returns = 2.939%
Q learner std deviation = 0.018
Nifty std deviation = 0.022

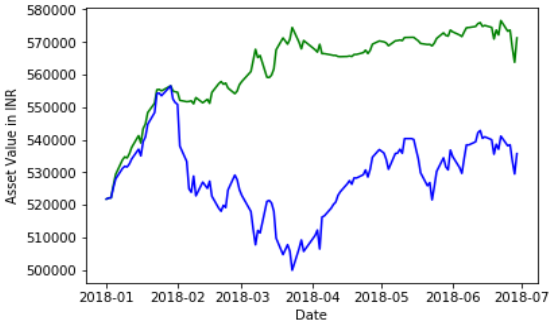


Q learner returns = 3.225%
Nifty returns = 2.939%
Q learner std deviation = 0.022
Nifty std deviation = 0.022

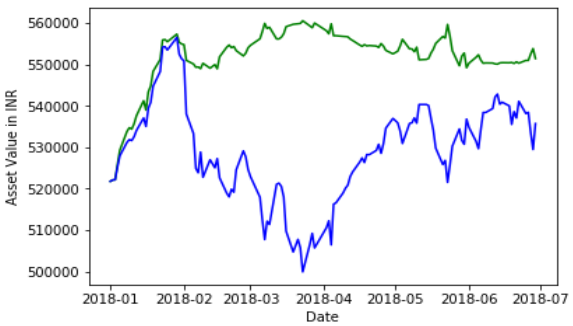


Q learner returns = 3.378%
Nifty returns = 2.939%
Q learner std deviation = 0.02
Nifty std deviation = 0.022

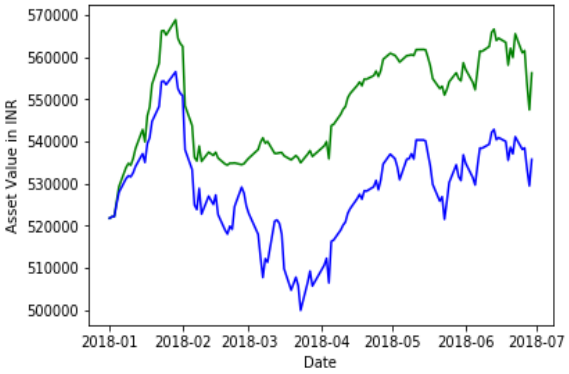
Quantity bought or sold for each trade = 5



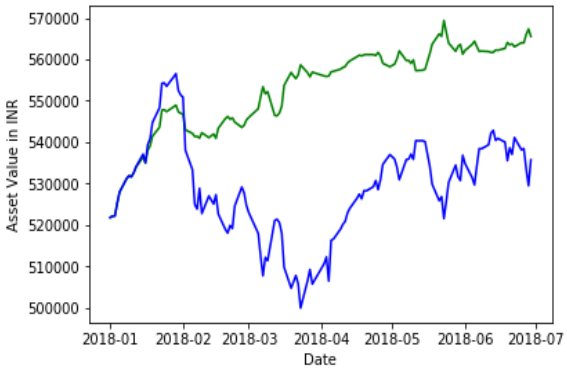
Q learner returns = 9.152%
Nifty returns = 2.939%
Q learner std deviation = 0.022
Nifty std deviation = 0.022



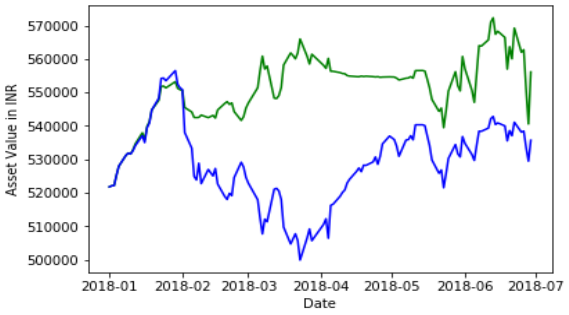
Q learner returns = 5.584%
Nifty returns = 2.939%
Q learner std deviation = 0.013
Nifty std deviation = 0.022



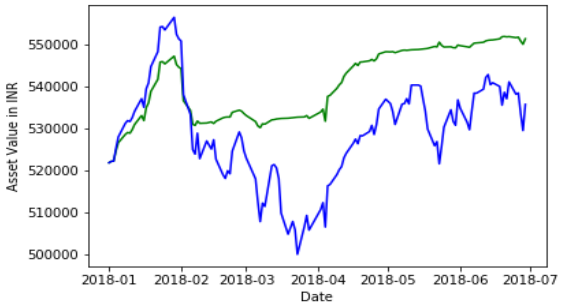
Q learner returns = 6.587%
Nifty returns = 2.939%
Q learner std deviation = 0.021
Nifty std deviation = 0.022



Q learner returns = 8.108%
Nifty returns = 2.939%
Q learner std deviation = 0.02
Nifty std deviation = 0.022



Q learner returns = 6.678%
Nifty returns = 2.939%
Q learner std deviation = 0.018
Nifty std deviation = 0.022



Q learner returns = 5.558%
Nifty returns = 2.939%
Q learner std deviation = 0.016
Nifty std deviation = 0.022

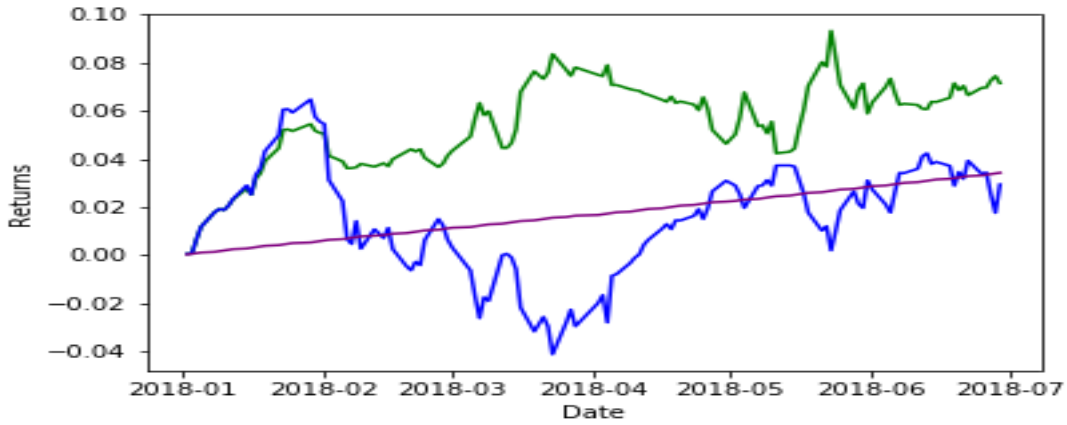
Justification

The returns I am getting in each case depends on the Q table which I am generating. The Q table is different for each instance of running the program because of the possibility of multiple actions for a particular state. For a state say (0, 0, 0, 1, 1, 1) corresponding to (5-day MA, 10-day MA, 20-day MA, VIX, RSI, MACD), 'buy' may give better returns for one case and 'sell' may give better returns for another case.

But for a higher trade size, the Q learner gives good returns on a consistent basis for all instances of running the program.

Conclusion

Free Form Visualization



Green represents the cumulative returns for Q learner

Blue represent the cumulative returns for Nifty index

Purple represents the cumulative returns for risk free bond

Here, Q learner returns = 7.141%

Nifty returns = 2.939%

Q learner std deviation = 0.018

Nifty std deviation = 0.022

Apart from giving significantly better returns, the Q learner always exhibits low standard deviation of the returns. So it gives good stable returns which are higher than both Nifty and risk free returns. This is mainly because Q learner takes the Sharpe ratio as reward for each trade. Since Sharpe ratio penalizes high volatility, the Q learner generates returns which are higher, but with low volatility. So it takes into account both returns and volatility while generating the Q table of state-action pairs

Reflection

The Q learning trading agent outperforms the Nifty index in majority of the cases. But it may sometimes underperform the Nifty index because of a different Q table. It is because of the possibility of multiple possible actions for a given state. In stock markets, multiple actions ('buy', 'sell', 'hold') may generate maximum risk adjusted return for a single particular state.

Another reason why the trading agent may underperform is that I lost some of the information present in the technical indicators since I transformed the technical indicators into discrete values.

Improvement

Some of the drawbacks of this Q learner may be removed by using a larger number of technical indicators which would increase the number of possible states. It can also be improved by increasing the in-sample size which would help the Q learner to come up with best possible action for a given state.

Another way would be to use deep Q learning which can make use of the continuous technical indicators instead of turning them into discrete values.

References

1. https://en.wikipedia.org/wiki/Q-learning
2. http://www.wildml.com/2018/02/introduction-to-learning-to-trade-with-reinforcement-learning/
3. http://www1.mate.polimi.it/~forma/Didattica/ProgettiPacs/BrambillaNecchi15-16/PACS_Report_Pierpaolo_Necchi.pdf
4. http://cs229.stanford.edu/proj2009/LvDuZhai.pdf
5. http://cs229.stanford.edu/proj2017/final-reports/5222284.pdf