Machine Learning Engineer Nanodegree

Capstone Proposal

Erik Flogvall
September 9th, 2017

Proposal

Domain Background

Machine learning is more increasingly used for predicting and trading stocks. One machine learning technique used to determine optimal trading policies is reinforcement learning. The iterative process of reinforcement learning is useful when the input data is added with every day. An automatic trading robot can be very useful if it can learn optimal policies for trading stocks. This should therefore be a very positive gain for any funds, stock brokers and individual currently not using reinforcement learning for this problem. The reinforcement learning has been in several scientific studies to make optimized and automatized traders [1][2].

I think that this is a interesting problem as it a case where a machine learning can use more information than what possible for a single person to take in to make the best trading decisions. I want to try Q-learning to make an automated trader. My belief is that Q-learning is a good method for this as it is model-free and relatively easy to understand when reading the Q-table.

Problem Statement

The main problem to be solved is to set up and train a reinforcement learner to output optimal actions for buying and selling stocks. The output of the problem will be what action of buying, selling or doing nothing is most beneficial for increasing the return from trading the stock.

The input data to this will the closing price for every trading day. A state for the stock will be created from the latest closing price and other values derived from historical closing prices (relative to the latest trading day). Examples of values that can be included in the state is the past daily return (daily change of the closing price), current and past closing prices and rolling statistical values of the closing price such as simple moving average [4] and moving standard deviation. The state also need to include if the stock is currently owned and information of any purchase price. Discretization will be necessary to create a discrete state as the closing price is a continuous input.

The size of the state should be kept as small as possible so it is possible to examine all states while training, but the state must also carry enough information. If there is too little information an optimal action cannot be selected. This will be one of the more challenging parts of this problem.

The return from trading using the optimal policy over a given test period will be used to make the problem measurable. This return will be compared to simply buying the stock at the start of the test

period and selling it and the end. If the reinforcement learning trader can make a consistently larger return than simply holding the stock over the test period is it working.

Datasets and Inputs

The dataset to be used in this capstone project is the Kaggle S&P 500 stock data collected by Cam Nugent under a public license [3]. The dataset contains stock prices in US dollar for all S&P 500 stocks for each trading day over five years. The included data in each entry is the price at the beginning of the trading day ("Open"), the highest price for each day ("High"), the lowest price for each day ("Low"), the closing price at the end of trading ("Close"), the number of traded shares ("Volume") and the stocks ticker name ("Name"). For this problem will only the closing price be used to build the state for the reinforcement learning

The data included in the dataset should be enough for making a discrete state for a given stock on a given date. A reward function can also be constructed by computing price changes. The size of the data set should be enough for both training and testing as the time series is relatively long (5 years) and there is data for 500 stocks.

Solution Statement

The proposed solution for the problem is to use Q-learning for learning an optimal trading strategy. The daily return for a stock should be the reward in the Q-learning algorithm. The possible actions in the Q-table should be buying, selling or doing nothing. After training should the Q-table contain information on what action is optimal for the given state. The result of the solution can be measured by the return from trading with the optimal policy.

Benchmark Model

The benchmark model will be buying the stock at the start of the testing period and selling it at the end. This is a non-active trading strategy commonly used for long term investment. It does not rely on daily stock information and should therefore be good for evaluating if the Q-learning trader can gain extra value by using more information.

Evaluation Metrics

To evaluate how well the model is working will the return over the trading period be computed. The total return over the trading period is defined as:

$$R_{TOTAL} = \sum R_i$$

Where R_i is the return for the i-th trade session, i.e. when stocks is bough hold for a given period and then sold. R_i is defined as:

$$R_i = S_i - B_i$$

Where S_i is income from selling the stocks, B_i is the cost of buying the stocks. The cost of trading the stocks is neglected to simply the problem.

Project Design

The main challenge is likely to be what should be included in the states. My strategy for making the states is to start with very simple states and gradually expand them by either increasing the number of discrete steps for each state variable or adding more state variables to add new information.

My proposed work flow is as following:

1. Divide the stock data in to two parts. One part for only training Q-learner and one part for continuing to train while also evaluating the Q-learner compared to the benchmark. This will simulate initializing the model on historical data and then testing it on real data.
    1. Divide the stock data into a training only part and a training+evaluation part. The training part should be the earliest 70% of the data and the training+evaluation data should be the last 30%. This means that the evaluation will only be done for the latest data and thereby simulate an implementation after training on historical data.
    2. Document in the report how the data was divided.
2. Create functionality for making the continuous stock data discrete.
    1. Write a function that creates bins based on the distribution of the training data that can be used for discretization.
    2. Write a function that sorts the continuous stock data into the bins to discretisize it.
    3. Document the method for discretization in the report.
3. Make a function for making discrete states for each day a stock is traded.
    1. Make a function that make states for the Q-table with the discretization functions. This will be done by sorting the training only data and sorting it to N bins so that each bin contains equal number of data points. The lowest and highest bin will have open sides interval to allow values larger and lower than what is in the training only data. This will discretisize the continuous variables into 0 to N-1.
    2. Write a draft section on the states are made in the report.
4. Implement the Q-learning algorithm. I want to try Dyna Q to improve the Q-learner between every trading day. The Q-learner should include discounted future rewards. The exploration rate will be decay over the training only part of the data and the learning rate will be constant.
    1. Make a function for initializing the Q-table
    2. Make a function selecting a the best action in the Q-table
    3. Make a function for executing the action, i.e. buy, sell or do nothing.
    4. Make a reward function from the change of the stock price the following trading day

5. Make learning function that updates the values in the Q-table with the value iteration update rule.
6. Make a function for running the the Q-learning process.
7. Document the how Q-learning algorithm was made in the report.

5. Implement a evaluation function.
    1. Make a function that return from the benchmark model-free
    2. Make a function that uses the learned Q-table to trade the stocks and evaluates the return from trading
6. Run the code and experiment with different types of states and tweak the q-learning algorithm.
    1. Select the method for making states and the tweaks to the code that worked best
    2. Update any changes in the report

7. Train the Q-learning algorithm on the training data
8. Run the trained Q-learning algorithm on the test data
9. Compare the results from running the Q-learning algorithm on the test data with the benchmark model on the same data.
10. Finalize the report.

References

[1] - https://www.cis.upenn.edu/~mkearns/papers/rlexec.pdf

[2] - http://cslt.riit.tsinghua.edu.cn/mediawiki/images/d/dd/%E6%B1%AA%E6%B4%8B-DQNStock.pdf

[3] - https://en.wikipedia.org/wiki/Moving_average#Simple_moving_average

[4] - https://www.kaggle.com/camnugent/sandp500