CS 4646
Project 8 | Strategy Learner
Joao Matheus Francolin

Introduction

In the final assignment for this class, we were tasked to create a strategy learner. We were given a choice to use three classes of machine learning algorithms; a classification-based learner, a reinforcement-based learner, or an optimization-based learner. After much consideration, I have chosen to model this trading problem as a reinforcement-based learner, more specifically, using an implementation of a Q-leaning algorithm. A reinforcement learning problem is such that an agent interacting with the world ought to take actions so that it attempts to maximize a notion of cumulative reward. Hence, in order to solve this reinforcement learning problem, we must define states, actions, and rewards. Notice that we don't need to construct a model of the environment itself. This is saves us enormous computing power, and It is one of the reasons RL is widely used on problems related to the stock market.

Rewards

One of the reasons that RL is a nice way to model trading problems is because the reward function can be easily defined as a smooth mapping of the trade returns. While there is more the one way to do such mapping, we want to construct a function that will give the agent rewards as often as possible, speeding up the rate of convergency. Since the granularity of the historical refers to market closing prices, the reward function was based on daily returns.

Actions

While there are many different types of orders accepted on a stock exchange, for the purpose of this project, we limited our universe to only BUY and SELL orders. Hence, I have defined the state space of the actions to be: LONG (100 shares), SHORT (100 shares), or HOLD (market neutral).

States

States a considerably more complicated to properly define on a trading problem. If we were to define the state of Mr. Pacman on maze, we would most likely use its grid coordinates, and we would certainty know where it stands in relation to itself and the rest of its environment. Stocks are different. Their only intrinsic value – their current price – does not inform us anything relevant regarding its location in relation to itself and the rest of the stock market. Instead, we must use secondary indicators that ultimately attempt to give us insight on the force and direction of the stock.

Three indicators were used in order to attempt to describe the state of the stock: Momentum, Moving Average Convergence Divergence (MACD), and Bollinger Bands. Because we do not wish to create any biases for our leaner, we must normalize the indicators so that they all have values

from the same range. Furthermore, the Q-learning algorithm requires discreate states and actions. Hence, we must also discretize the states

- **Normalization**
  Create a normal distribution where most values range between -1 and 1.

  $For\ each\ indecator$:

  $$norm\_indicaotor = \frac{indicator - indicator.mean()}{indicator.std()}$$

- **Discretization**
  Group the data into bins according to its density on the distribution.
  $For\ each\ indecator$:

  $Define\ number\ of\ bins\ (10\ in\ this\ implemantation)$
  $Sort\ data$
  $Divide\ data\ on\ equal\ numbered\ (but\ differently\ sparsed)\ bins$

  We now have four sets of discretized numbers (Bollinger Bands have one value for an upper band, and another for a lower band) ranging from 0 to 9. In order to train the Q-learner, we concatenate these numbers into a single integer, and feed it to the learner as a state.

Experiment 1 – Learned Strategy v. Manual Strategy

The last trading strategy developed in this class was a manual strategy based on the same indicators discussed above. The difference, of course, was that we, the students, were tuning the hyper-parameters manually, and arbitrarily creating a logic to generate trading signals. My manual logic was design was follows:

- **Manual Rule**
  Given the indicators Momentum, MACD, and Bollinger Bands, I created a signals data frame with corresponding column for each indicator. The only allowable values on the signal data frame were: -1 (SHORT), 0 (NEUTRAL), 1 (LONG). The columns were updated according to the following logic:

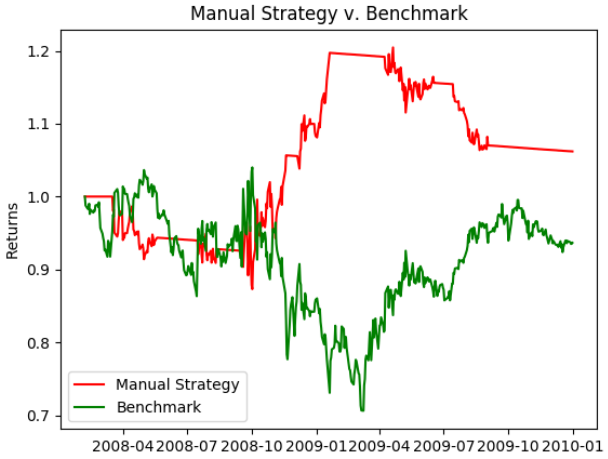
$If\ \text{MOMEMTUM} > 0.8 \rightarrow \text{LONG}$
$If\ \text{MOMEMTUM} < 0.5 \rightarrow \text{SHORT}$

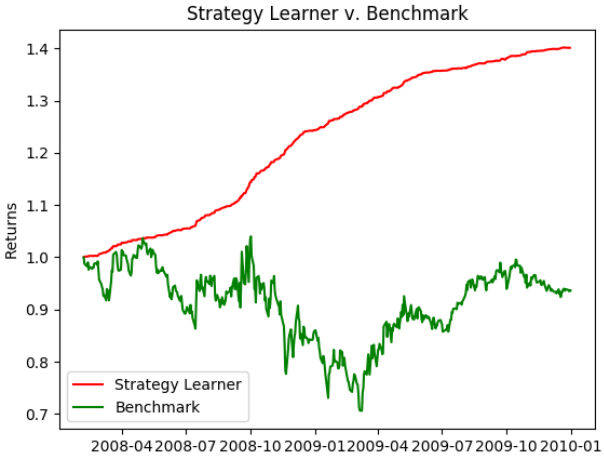
$IF\ MACD > 0.5 \rightarrow LONG$
$IF\ MACD < -0.5 \rightarrow SHORT$

$If\ Price < \text{BB}_{\text{Lower Band}} \rightarrow \text{LONG}$
$If\ Price > \text{BB}_{\text{Upper Band}} \rightarrow \text{SHORT}$

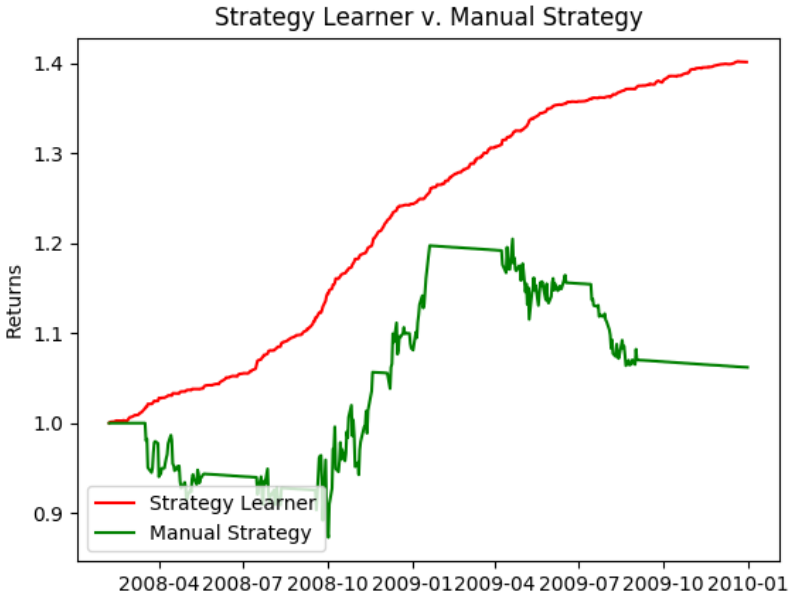
After the logic is apply, the indicators vote. If the vote to go LONG is a majority, a LONG order is created. Likewise, If the vote to go SHORT is a majority, a SHORT order is created. If no consensus is found, a HOLD order is created. This it will exit any possible open positions.

The manual rule performed readably well on the in-sample data. When it is plotted against the benchmark, a perpetual LONG 100 shares of the JPM stock, it becomes clear that the strategy took advantage of the downward movement of the stock. The manual strategy, however, does to seem to have a consistent behavior. The gains were very dependent on the abrupt fall in price of the stock, and when the stock started gaining traction again, the strategy started performing poorly. In the other hand, we can see that the strategy learned by the reinforcement leaning agent was very consistent. It had similar rate of daily returns all through the in-sample data period; independent of the stock price underlining behavior



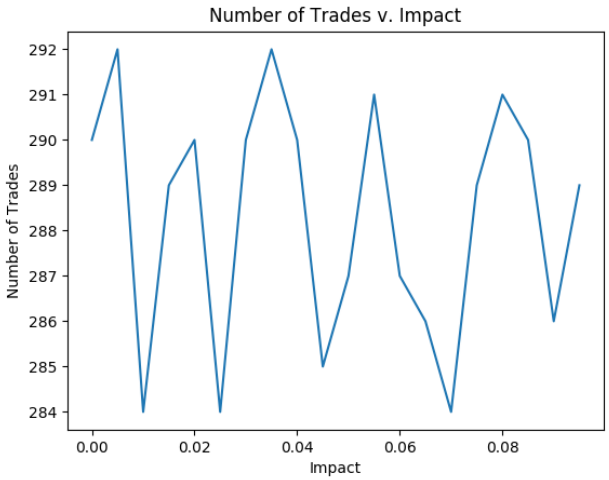


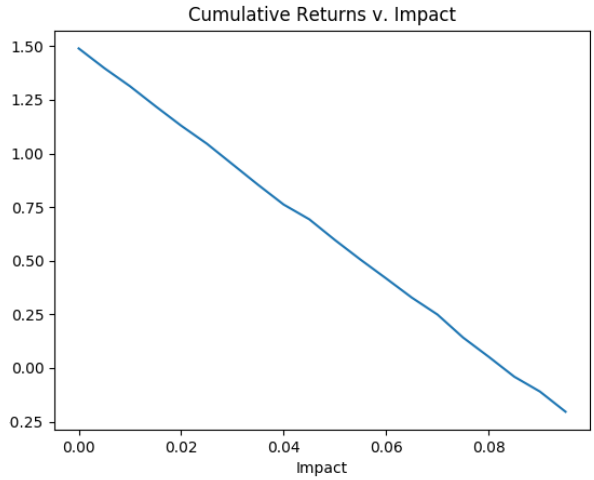
The advantage of the strategy learned by the reinforcement learning agent becomes clearer when we directly comparing the results of both strategies. The q-leaning agent had consistent gains, with a greater overall return and less volatility when compared to the manual rule agent.



Experiment 2 – Agent response to changes on market impact

On this experiment we trained and tested the reinforcement learning agent under different market conditions. More specifically, the again was re-trained and tested 20 times. For every new test, the impact in which the agent had on the market when performing a trade was increased by 0.005 points, ranging from 0% to 10% on the scope of the experiment.





The first hypothesis considered was that the agent was that the agent would decrease the rate in which it would perform trades, hence performing only the very best trades. We see that this hypothesis was confirmed. Regarding the number of trades, we can verify that the trading patters changes were uncorrelated to the increase in market impact. Regarding to the selection of trades, we see that better trades were not selected given that returns were always in a downward slope.

The second hypothesis was that the agent was inevitably going to perform worse. This was expected because robot traders (in general) take advantage of very small divergencies on price trends, and therefore they must perform as many trades as possible. Increasing the impact it had on the market handicapped the agents ability to profit on small spreeds.