Syed Ahmed
GTID: sahmed99
Course: ML4T

Strategy Learner

Introduction:

In this assignment we were required to build a strategy learner using one of the machine learning algorithms learned in this class. My choice of algorithm was using random forest created by bagging RTLearner. This assignment was broken into four tasks. The first task was to implement addEvidence() which is a function to train learner using two Bollinger Band and Simple Moving Average indicators. Note: these indicators were chosen because they were previously used in Manual Learning Strategy. Second task was to implement testLearner() which would test the learner on a new set of data. Third task was to perform an experiment comparing results of Manual Strategy, Benchmark Strategy and Strategy Learner to see which performs the best. Lastly, the fourth task was to observe any change that was caused by increasing impact.

How The Learner is Utilized:

The strategy learner is utilized by first learning through the classifier when to buy “1”, sell “-1” and hold “0”. The learner learns through two data sets trainX and trainY. The dataset trainX contains the values for the indicators for each day, while trainY contains buy or sell for each day. Once the data set is created, it goes to bagLearner which creates 20 instances of RTLearner, once the instances are created, the learner can be queried using just the indicator values (trainX), this returns closely adjusted values from our RTLearner which I used to create a trading strategy for what days to buy, sell and hold.

Learner Set Up:

To implement the Strategy Learner API, I used BagLearner as the learner. The BagLearner uses RTLearner and take a leaf_size of 5 and bag size of 20. The BagLearner creates 20 instances of RTLearner.

No adjustment of data was needed because RTLearner did not require it. Ensemble technique of baglearner with RTLearner was used to teach the learner about the data. Data was classified as X and Y for labels.

Implementing addEvidence()

To implement addEvidence() I first obtained the prices of the given symbol and date, next I obtained Bollinger Band and Simple Moving Average indicators and set their value as my X-train dataset. To obtain the Y-train values, I looped through the price data and obtained a ratio value which was derived by taking price[i + Nday] - price[i] / price[i]. This ratio was compared with market variance of 2%, if the ratio was above market variance + impact then that signaled a Buy or "1", if it was below than that signaled a sell or "-1". If the ratio was neither greater than or equal to then that signaled a hold opportunity. I used the signals obtained to construct my Y-train dataset.

Implementing testPolicy()

I implemented testPolicy() by first obtaining the price for given date, then obtaining the Bollinger Band and Simple Moving Average indicators and using their values as my testX data set. Next, I queried my learner and obtained the resultY values which were between -1 and 1. By iterating through these values, I created my trade list, if value[i] was greater than 0 that signaled 1000, if it was less than 1 that signaled sell. The trade holding was restricted to be between +2000 to -2000.

Experiment 1:

For experiment 1, I compared the portfolio value produced from strategy learner, manual learner and bench mark. Bollinger Band and Simple Moving average indicators were used for strategy learner and manual learner. The manual strategy code algorithm uses metric obtained from indicators to derive trade actions, while strategy learner uses ensemble machine learning to derive trade actions. Bench mark is simply buying 1000 shares at the start date and selling at the end date.

Assumptions:

For experiment 1, commision and impact were kept at 0 while we had unlimited leverage.

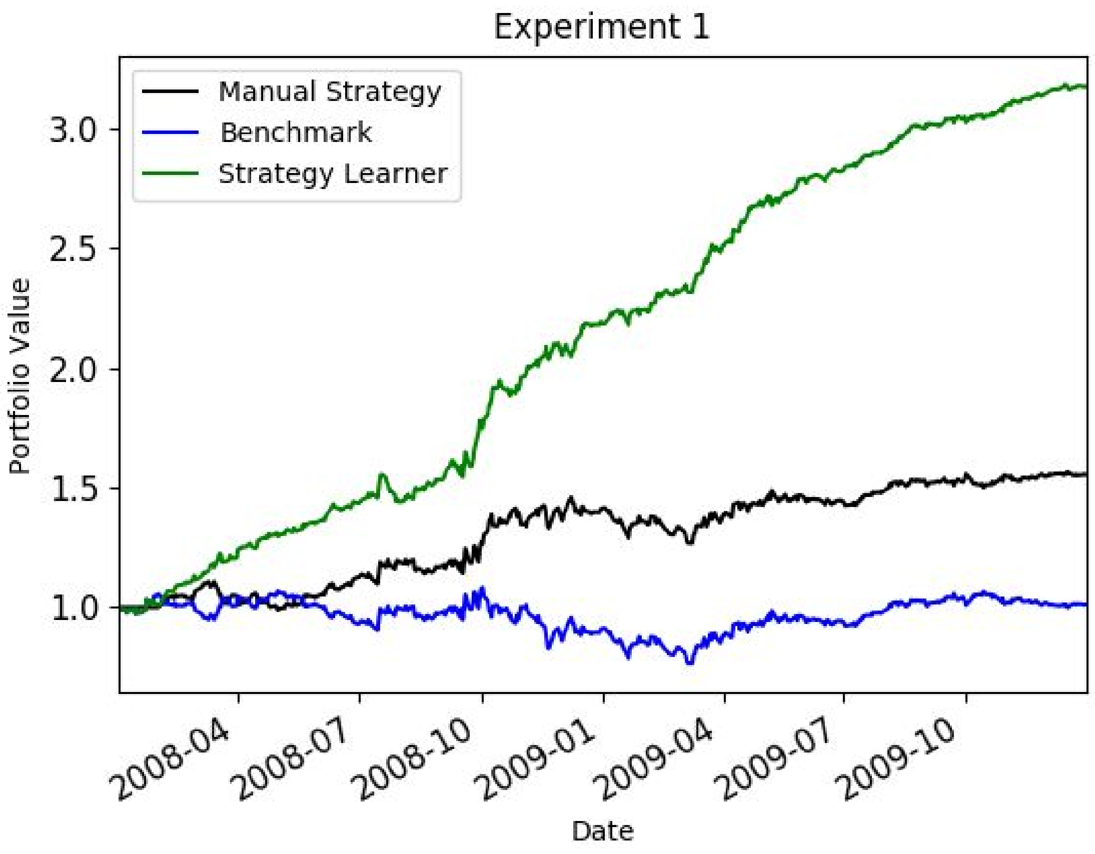
Parameter:
Symbol: JPM
In-sample period: January 1, 2008 to December 31, 2009
Starting cash: $100,000
Allowable positions: 1000 shares long, 1000 shares short, 0 shares
Commission: $0.00
Impact: 0.00
Leaf-size: 5
Bag-size: 20

Program flow:

1. Create strategy learner object, with the given parameters and call addEvidence() - This will train the learner. Once the training is complete, call testPolicy() which will query the in sample data and return a list of trades.
2. These trades are then put used by marketsim compute_portval() which returns the portfolio value for the list of trades.
3. Call the benchmark function from TheoreticallyOptimalStrategy and perform step # 2.
4. Call the testPolicy from ManualStrategy and perform step #2.
5. Take all the portfolio values from the given algorithms and plot them.

Experiment 1 results:
It can be observed from the chart below that strategy learner cumulative return value is greater than both strategy learner and bench mark. The difference is quite significant, but expected since only in sample data was used for learning and testing. If only testing with in-sample data, I would expect strategy learner to constantly perform better than manual strategy and benchmark since the algorithm is learning the data and making predictions based on the same data. However if out of sample data is used then results may differ and manual strategy may perform better but I believe strategy learner will still outperform benchmark.

Portfolio Stats	Benchmark	Manual Strategy	Strategy Learner
Cumulative Return	0.0123	0.555	2.17
Daily return STD	0.016	0.012	0.0089
Daily return Mean	0.00016	0.0009	0.00223
Sharpe ratio	0.1567632	1.2212525118	3.72017



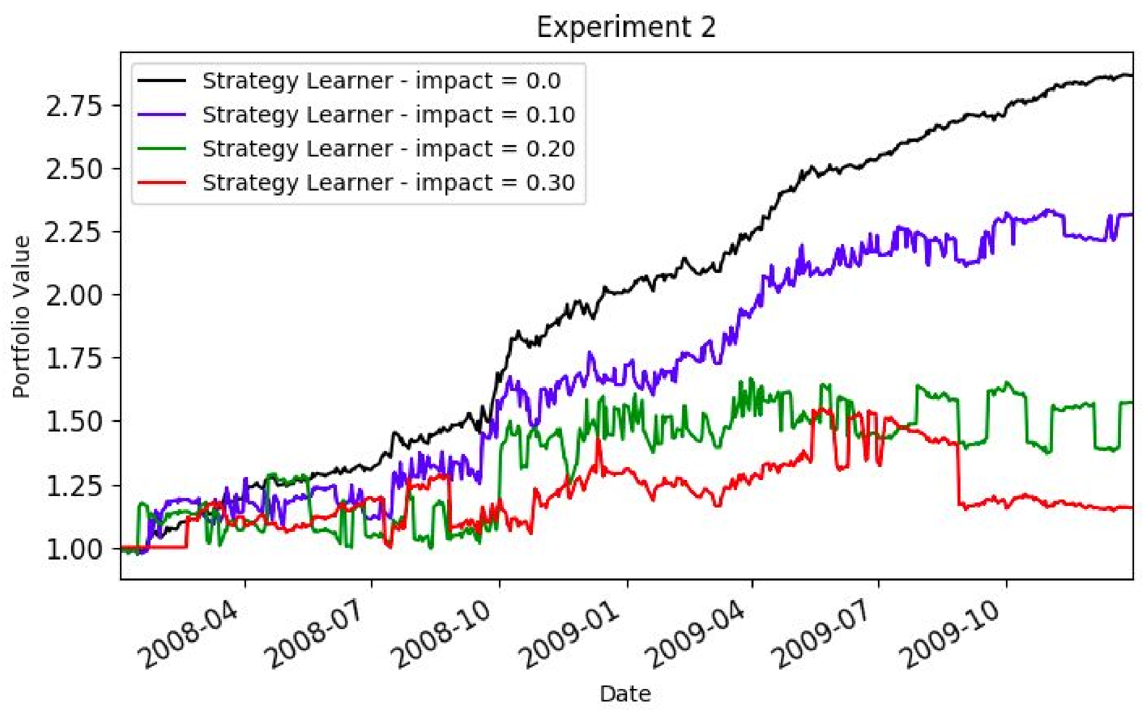
Experiment 2:

In experiment 2, I changed the impact value for strategy learner to see how it affects the performance of the portfolio. My hypothesis is that a higher impact value will lead to a lower return since impact is the cost that a buyer or seller incurs while executing a transaction due to the prevailing liquidity conditions. I will be using impact value of 0%, 10%, 20%, and 30% for my observation. For my metrics, I have used the cumulative return and number of trades for each impact value. I conducted experiment using JPM stock symbol and in sample period to test the hypothesis.

Program flow:

1. Create strategy learner object with impact of 0. Train learner with in sample data and test with in sample data.
2. Take the trades list and calculate number of trades for impact value and cumulative return.
3. Repeat step 1 and 2 for impact value 10%, 20% and 30%.
4. Plot and print results

Experiment 2 results:
As predicted, the number of trades along with portfolio performance dropped with increase in impact value.



Impact	Number of trades
0%	173
10%	138
20%	72
30%	16

Conclusion:

In strategy learner assignment we developed a machine learning strategy using random forest RTLearner. This allowed us to learn from in sample data to predict future buy or sell opportunities based on Bollinger Band and Simple Moving Average indicators. In experiment one, this strategy was compared with manual strategy and bench mark, where for in sample data it outperformed both, however more tests need to be conducted on out-sample data to make actual bets with real money. The market impact is also a big unknown factor and it has great effect on the performance of strategy so it also needs to be taken into account to create a formidable trading strategy.