CS 7646 – Machine Learning for Trading (Fall 2017)

Juan Carlos Kuri Pinto, GTid: 903271649
jckp3@gatech.edu, jckuri@gmail.com

Project 8: Strategy Learner

Part 1: Technical Indicators and Opto Trader

This strategy uses 3 technical indicators:

1. Moving Sharpe Ratio
2. Bollinger Bands
3. Momentum

Indicator 1: Moving Sharpe Ratio

The Sharpe ratio is a commonly used measure of risk-adjusted return:

$$SR = \sqrt{samplings} \cdot \frac{\bar{r}_p - r_f}{\sigma_p} \quad (1)$$

Assuming the risk-free rate is zero ($r_f = 0$) and the square root of samplings is hidden by the z score, the Sharpe ratio can be rewritten as:

$$SR = \frac{\bar{r}_p}{\sigma_p} \quad (2)$$

Which is composed of the average and the standard deviation of the portfolio return. Given the enormous success of the Sharpe ratio, why not making it an indicator for financial time series? That can be done by computing the daily returns of the portfolio first. Then, the Moving Sharpe Ratio (MSR) can be computed as the moving average (rolling mean) of the daily returns divided by the moving standard deviation (rolling std) of the daily returns. The result was truly awesome because MSR was the most relevant of the 3 indicators used in this project. The MSR can inform the trader when there are periods of steady growths and steady declines.

Python code:

```
# moving sharpe ratio
def indicator_1(port_val, days):
    daily_rets = (port_val / port_val.shift(1)) - 1.0
    daily_rets = daily_rets[1:]
    rr = daily_rets.rolling(window = days, center = False)
    sr = rr.mean() / rr.std()
    return get_z_score(sr)

def get_z_score(df):
    return (df - df.mean()) / df.std()
```

In all the 3 indicators, the moving window for the rolling statistics is 30 days. Other periods of time do not produce better results. Why? Perhaps because months have 30 days and human activities are discretized by months.

Indicator 2: Bollinger Bands

The Bollinger Bands (BB) is a famous indicator which uses the rolling mean in the center of a financial time series and 2 bands which are 2 standard deviations above and below the rolling mean. BB can inform the trader when there are buying and selling opportunities.

Transforming BB into an indicator is relatively easy. Just compute the rolling mean first. Then compute the rolling standard deviation. The difference is the price of the stock minus the rolling mean. BB is the difference divided by 2 rolling standard deviations.

Under this mathematical definition, if BB goes above 1 and returns below 1, it is a selling opportunity because a high price will decline to its trend (the rolling mean). And if BB goes below -1 and returns above -1, it is buying opportunity because a low price will increase to its trend (the rolling mean). BB reaches its highest points when the price goes beyond the upper band. BB reaches its lowest points when the price falls beyond the lower band.

Python code:

```
# bollinger_bands
def indicator_2(port_val, days):
    r = port_val.rolling(window = days, center = False)
    rm = r.mean()
    rs = r.std()
    bb = (port_val - rm) / (2 * rs)
    return get_z_score(bb)
```

Indicator 3: Momentum

Momentum is yet another famous indicator that informs the trader when there are events of significant growths or significant declines. Thus, momentum indicates when it is appropriate to buy or to sell, respectively.

Momentum reaches its highest points when there are periods of significant growth. Momentum reaches its lowest points when there are periods of significant decline. Calculating momentum is trivial. Just divide the price of the stock by its price some days ago. (30 days.) Subtract 1 to convert it into a percentage of growth.

Python code:

```
# momentum
def indicator_3(port_val, days):
    momentum = port_val / port_val.shift(days) - 1.0
    return get_z_score(momentum)
```

Here is a graph with the 3 indicators overlapped. They are somewhat correlated. But in the Opto Trader, Moving Sharpe Ratio (Indicator 1) is very negatively correlated with the linear frontiers.

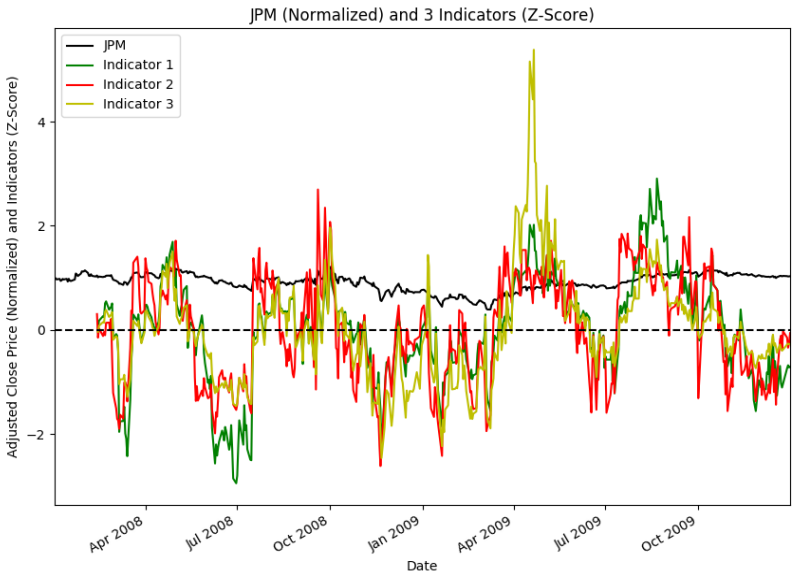

Figure 1: The 3 indicators overlapped

In this graph, there are 2 types of curves: **One type of curve** is adjusted close price of JPM for the in-sample period. This curve is normalized by the initial value, making it begin with one. Regarding normalized prices, one means 100% of the initial value; 1.5 means 150% of the initial value; -0.5 means -50% of the initial value; and so on. **And the other type of curve** is z-score of the indicators. Regarding the z-score, zero means the mean; 1 means 1 sigma above the mean; -2 means 2 sigmas below the mean; and so on.

Opto Trader

Basically, Opto Trader is made of 2 tuned linear combinations of the 3 indicators explained above:

$$if\left(w_{1a}\cdot indicator\,1+w_{2a}\cdot indicator\,2+w_{3a}\cdot indicator\,3<-1\right)then\;goal=goal+1 \tag{3}$$

$$if\left(w_{1b}\cdot indicator\,1+w_{2b}\cdot indicator\,2+w_{3b}\cdot indicator\,3>1\right)then\;goal=goal-1 \tag{4}$$

At the beginning, goal is 0. Both conditions can be false, leaving goal = 0. Both conditions can be true, summing up goal = 0. Or only one condition can be true, making goal be +1 or -1. Then, “goal of shares” = goal * 1000. Trade, goal of shares, and current shares operate with the mechanism of equation (5). The algorithm suggests a goal of shares: +1000, 0, or -1000 shares. But the actual shorts or longs are computed with this trade formula:

$$trade=goal\;of\;shares-current\;shares \tag{5}$$

By using trial-and-error, one can vary the weights w and examine the overall performance of the portfolio by using the market simulator code of a previous project. The cumulative return is the variable to optimize. Prof. Balch suggested to use Sharpe Ratio as the variable to optimize. But cumulative return as the variable to optimize produced better results because one of the indicators already uses Sharpe Ratio. Thus, Sharpe Ratio as the variable to optimize is redundant.

This tedious process consists in trying different values and increments in values for the weights and selecting those weights whose cumulative return is the biggest. This process is very similar to evolution by natural selection.

Since this process is really boring and slow, a trial-and-error search was programmed in which each dimension is incremented at each step with the following values: 0.0, -0.01, -0.05, -0.1, -0.5, -1.0, -2.5, -5.0, -10.0, 0.01, 0.05, 0.1, 0.5, 1.0, 2.5, 5.0, 10.0. There are 3 loops: The main loop which stops at convergence, aka no further improvement. The dimensional loop that iterates each dimension. And the increment loop that varies the small increments in each dimension and selects the increments with the most rewarding cumulative return.

At the end of the trial-and-error search, the following weights were found which produce a cumulative return of 270.61% in the in-sample period of JPM. These weights correspond to commission=$9.95 and impact=0.005.

$$\begin{matrix} w_{1,a}=-5 & w_{2,a}=1.5 & w_{3,a}=2.45 \\ w_{1,b}=-11.05 & w_{2,b}=4.7 & w_{3,b}=3 \end{matrix} \tag{6}$$

Indicator 2 (Bollinger Bands) and indicator 3 (Momentum) are useful and positively correlated with the linear boundaries above. Indicator 1 (Moving Sharpe Ratio) is **very negatively** correlated with the linear boundaries, making it the most useful indicator in this project.

Part 2: Experiment 1

In a previous project Manual Strategy, the same code of Opto Trader was used and the results were truly awesome. However, there was a taste of cheating when doing the trail-and-error search with a Python script. For the sake of fairness, here are the weights w that were found in a manual way without cheating, that is, without automatizing the process of trial-and-error search:

$$\begin{matrix} w_{1,a}=-2 & w_{2,a}=1 & w_{3,a}=2 \\ w_{1,b}=-2 & w_{2,b}=1 & w_{3,b}=1 \end{matrix} \tag{7}$$

By using commission=$9.95 and impact=0.005, these weights produced the following results in the Manual Strategy. And the weights in equation (6) produce the following results in the Opto Trader. The results in Benchmark are obtained by buying 1,000 shares of JPM at the beginning of the period and holding them until the end.

	BENCHMARK (In Sample)	**MANUAL STRATEGY (In Sample)**	**OPTO TRADER (In Sample)**
Sharpe Ratio	0.156918406424	4.16624009415	5.73119625052
Cumulative Return	0.0123	1.454789	2.7060705
Standard Deviation of Daily Returns	0.0170043662712	0.00688432704927	0.00728127695248
Average of Daily Returns	0.000168086978191	0.00180678101113	0.0026287698189
Portfolio Value	101230.0	245478.9	370607.05

Table 1: Performance comparison of 3 strategies

Machines are definitely better than humans when doing tedious computations. The weights found by the computer are more finely tuned, thoughtful, and have sharper synergies and small

granularity. Whereas the weights manually tuned do not have decimals (big granularity) and are unfit, less risky, and shorter. However, both the weights found by computer and the weights manually tuned pass all the test cases of Strategy Learner in a consistent way.

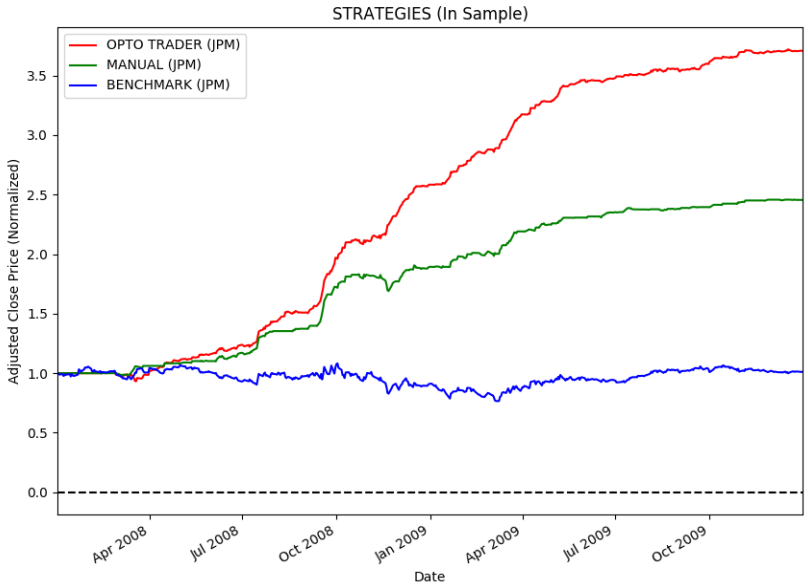

Figure 2: Performance of the 3 strategies

These relative results are always expected in all kinds of stocks and periods (in-sample or out sample). Why? Because the 3 indicators used are very informative and general for financial data. These relative results can differ a little bit if the dataset is specially designed to fool the 3 indicators. If the period is smaller than the window period of 30 days, Opto Trader won't work at all. Extreme values of impact also disable Opto Trader as seen in the next section.

Part 3: Experiment 2

The weights found in equation (6) are insensitive to different values of impact. However, there is a clever way to make Opto Trader sensitive to different values of impact. One can run an even bigger meta-optimization process in which different weights are specially adapted to different values of commission and impact.

Such meta-optimization process ran for more than 20 hours in the Buffet machines. However, this huge amount of time spent had a positive consequence: The function addEvidence() of StrategyLearner is computed in 0 seconds because it only initializes objects with some values. Remember that the function addEvidence() is supposed to run in less than 25 seconds. The meta-optimization process ran Opto Trader many times. How many times? As many times as the cardinality of the Cartesian product of these 2 lists of values:

commissions = [0.0, 9.95]

impacts = [0.0, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1.0, 5.0, 10.0, -0.001, -0.0025, -0.005, -0.0075, -0.01, -0.025, -0.05, -0.075, -0.1, -0.25, -0.5, -0.75, -1.0, -5.0, -10.0]

$$|commissions \times impacts| = 2 \cdot 31 = 62 \text{ times} \quad (8)$$

The optimization process of Opto Trader was run 62 times. Each run lasted many minutes. That's why it took more than 20 hours to fill a Python dictionary whose keys are (commission, impact) and whose values are the optimal weights found for each combination of (commission, impact). Such big dictionary of weights was hardcoded in Opto Trader, so that each call to the function testPolicy() will search for the most similar combination of (commission, impact) and will use the most appropriate weights to generate the trades. This method allowed Opto Trader to be sensitive to different values of commission and impact.

In general, positive values of impact decrease the trader's amount of money. That's why impact is usually small, like 0.005. Otherwise, if the impact would be much higher like 0.50 (50%), each transaction would be too expensive for the trader. Impact is **like** a tax. If taxes are too big, transactions are discouraged. That's why many rightist politicians prefer to diminish taxes. However, taxes are important to finance public entities and public services, as leftist politicians suggest. Therefore, the correct amount of taxes is a trade-off to be debated and to be balanced.

Negative values of impact are rare. It's like if governments were encouraging traders to make more transactions. There are some countries in which negative taxes exist. For example, some governments give money to students and Universities to encourage education or give money to pregnant women to increase the natality rate in aging populations. In this particular case, negative values of impact were considered for two reasons: First, for the sake of completeness to explore the behavior of the following graphs in a more comprehensive way. Second, to try to successfully pass the test cases of the autograder.

In Figure 3, the number of trades is analyzed. At impact=0, the amount of trades is about 230. Negative values of impact slightly increase the amount of trades. And positive values of impact significantly decrease the amount of trades. At impact=0.10 (10%), the amount of trades drops close to 0. That's why a default value of impact should be 0.005 (0.5%).

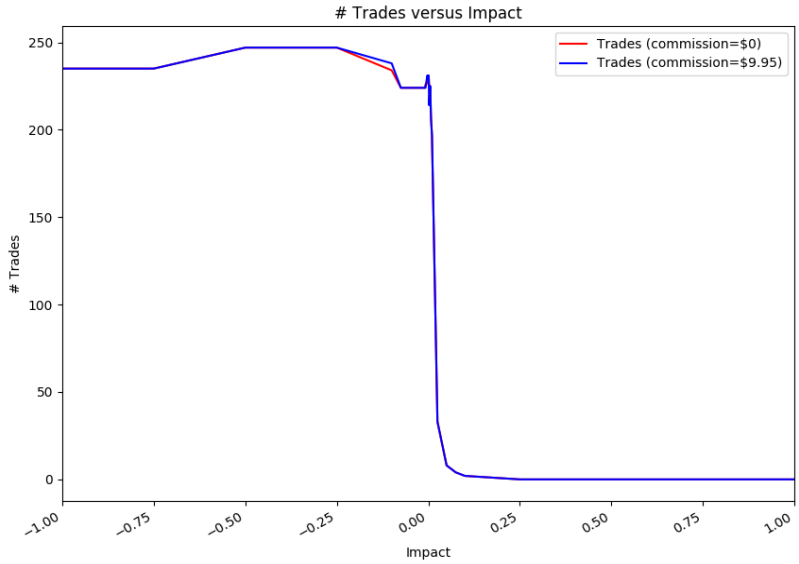

Figure 3: How the number of trades varies with different impact values

Notice that Opto Trader is somewhat insensitive to different values of commission: $0 and $9.95. The red curve of commission=$0 and the blue curve of commission=$9.95 are very overlapped. Don't be confused. Opto Trader is sensitive to different values of commission. Try to see how the red curve slightly escapes the overlapping at impact=0.20 approximately. Why are both curves overlapped? Because a commission of $10 is insignificant with respect to transactions of 1000 or 2000 shares, and each share costs many dollars. The same happens in the 3 graphs of Experiment 2.

In Figure 4, the cumulative return (CR) is analyzed. At impact=0, the CR is decent, between 2 and 3. However, negative values of impact dramatically improve CR. Impact=1 produces a huge CR of 100+, which is absurd. On the other hand, positive values of impact slightly decreases CR, until CR asymptotically converges to 0. Too much impact disallows earnings, which is also absurd because nobody will invest in such hypothetical market.

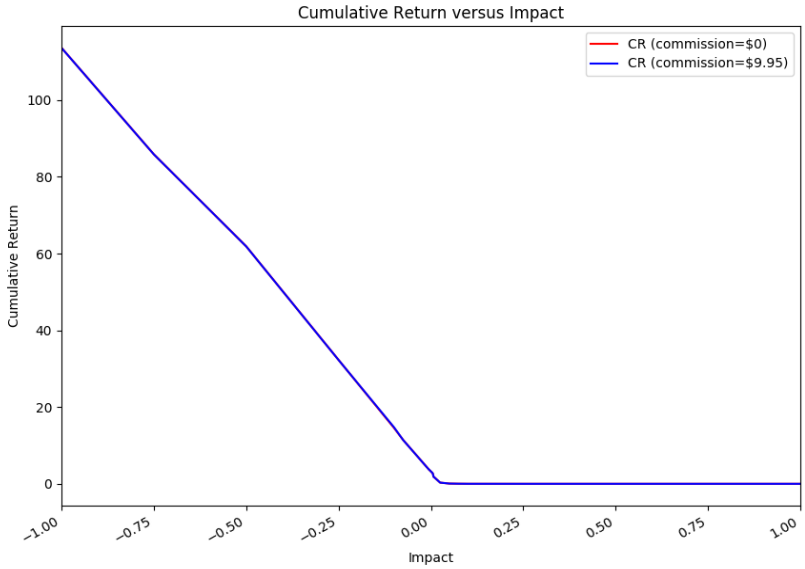

Figure 4: How the cumulative return varies with different impact values

In Figure 5, the magnitude of weights is analyzed. Remember that weights balance the participations of the 3 financial indicators to decide when to buy and when to sell, if certain thresholds are triggered. Remember also that these weights are optimized in order to maximize CR. At impact=0, weights have a magnitude of 10 approx. Negative values of impact make the weights grow, triggering more buys and more sells. Positive values of impact make weights shrink, making buys and sells less frequent.

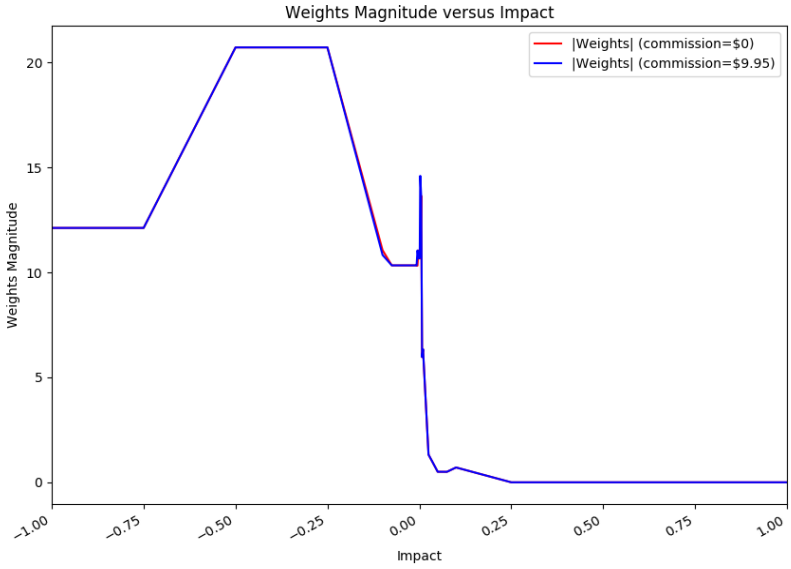

Figure 5: How the weights magnitude varies with different impact values