Summer 2016 Project 4

From Quantitative Analysis Software Courses

Contents

FAQs

- Q: In a previous project there was a constraint of holding a single position until exit. Does that apply to this project? Yes, hold one position until exit.
- Q: Is that 5 calendar days, or 5 trading days (i.e., days when SPY was traded)? A: Always use trading days.
- Q: Are there constraints for Python modules allowed for this project? Can we experiment with modules for optimization or technical analysis and cite or are we expected to write everything from scratch for this project as well? A: You can use scikit modules as long as you cite them.. You've already written the learners you need though.
- Q: Can we change our policy to work better for IBM vs the sine data? A: No, you must use the same indicators, policy, etc. for both. I suggest you optimize first for IBM, then go back to the sine data because almost anything should work with the sine data.
- Q: I want to read some other values from the data besides just adjusted close, how can I do that? A: Please modify an old version of util.py to do that, include that new util.py with your submission.
- Q: Are we required to trade in only 100 share blocks? (and have no more than 100 shares long or short at a time as in some of the previous assignments) A: Yes. This will enable comparison between results more easily.
- Q: Are we limited to leverage of 2.0 on the portfolio? A: There is no limit on leverage.
- Q: Are we only allowed one position at a time? A: You can be in one of three states: -100 shares, +100 shares, 0 shares.

- Q: Are we supposed to build one policy that we use on both SINE and IBM? A: Yes, all parameters for your learner and policy should be the same. The difference is the DATA.

Overview

In this project you will transform your regression learner into a stock trading strategy. You should train a learner to predict the change in price of a stock over the next five trading days (one week). You will use data from Dec 31 2007 to 2009 to train your prediction model, then you will test it from Dec 31 2009 to 2011.

Now, just predicting the change in price isn't enough, you need to also code a policy that uses the forecaster you built to buy or sell shares. Your policy should buy when it thinks the price will go up, and short when it thinks the price will go down. You can then feed those buy and sell orders into your market simulator to backtest the strategy. For ease of comparison between strategies, please observe these rules:

- Starting cash is $10,000.
- Allowable positions are: 100 shares long, 100 shares short, 0 shares.
- There is no limit on leverage.

Finding features, a learner, and a policy that all work together to provide a reliably winning strategy with live stock data is HARD! It is possible, and people have done it, but we can't reasonably expect you to be successful at it in this short class. Accordingly, we want you to work with some easy data first, namely we will provide you with sinusoidal historical price data. Once you've got something that works with that, you can try your learner on real stock data.

Detailed steps

Overall, you should follow these steps:

- Train a regression learner (KNN or LinReg, or other of your choice with or without bagging) on data from Dec 31 2007 to Dec 31 2009. This is your in sample training data.
  - For your X values: Identify and implement at least 3 technical features that you believe may be predictive of future return. You should implement them so they output values typically ranging from -1.0 to 1.0. This will help avoid the situation where one feature overwhelms the results. See a few formulae below.
  - For your Y values: Use future 5 day return (not future price). You're trying to predict a relative change that you can use to invest with.
- Create a plot that illustrates your training Y values in one color, current price in another color and your model's PREDICTED Y in a third color. To help with the visualization, you should adjust your training Y and predicted Y so that they are at the same scale as the current price. With this chart we should be able to see how well your learner performs and that your Y values are shifted back 5 days. You may find it convenient to zoom in on a particular time period so this is evident.
- Create a trading policy based on what your learner predicts for future return. As an example you might choose to buy when the forecaster predicts the price will go up more than 1%, then hold for 5 days.
- Create a plot that illustrates entry and exits as vertical lines on a price chart for the in sample period Dec 31 2007 to Dec 31 2009. Show long entries as green lines, short entries as red lines and exits as black lines. You may find it convenient to zoom in on a particular time period so this is evident.
- Now use your code to generate orders and run those orders through your market simulator. Create a chart of this backtest. It should do VERY well for the in sample

period Dec 31 2007 to Dec 31 2009.
- Freeze your model based on the Dec 31 2007 to Dec 31 2009 training data. Now test it out of sample over the period Dec 31 2009 to Dec 31 2011. Create a plot that illustrates entry & exits, generate trades, run through your simulator, chart the backtest.

Perform the above steps first using the data ML4T-240.csv. Once you've validated success (it should work well), repeat using IBM data over the same dates. Remember Dec 31 2007 to Dec 31 2009 is training, Dec 31 2009 to Dec 31 2011 is testing. You should have one set of charts for each symbol.

Summary of Plots To Create

1. Sine data in-sample Training Y/Price/Predicted Y: Create a plot that illustrates your training Y values in one color, current price in another color and your model's PREDICTED Y in a third color. To help with the visualization, you should adjust your training Y and predicted Y so that it is at the same scale as the current price.
2. Sine data in-sample Entries/Exits: Create a plot that illustrates entry and exits as vertical lines on a price chart for the in sample period. Show long entries as green lines, short entries as red lines and exits as black lines. You may find it convenient to zoom in on a particular time period so this is evident.
3. Sine data in-sample backtest
4. Sine data out-of-sample Entries/Exits: Freeze your model based on the in-sample data. Now test it for the the out-of-sample period. Plot the entry & exits, generate trades,
5. Sine data out-of-sample backtest.
6. IBM data in-sample Entries/Exits: Create a plot that illustrates entry and exits as vertical lines on a price chart for the in sample period 2008-2009. Show long entries as green lines, short entries as red lines and exits as black lines. You may find it convenient to zoom in on a particular time period so this is evident.
7. IBM data in-sample backtest
8. IBM data out-of-sample Entries/Exits
9. IBM data out-of-sample backtest

Template and Data

You should create a directory for your code in ml4t/p4. You will have access to the data in the ML4T/Data directory but you should use ONLY the code in util.py to read it. In particular files named ML4T-240.csv, and IBM.csv.

Choosing Technical Features -- Your X Values

Here's a suggestion of how to normalize Bollinger Bands so that feature so that it will typically provide values between -1.0 and 1.0:

```
bb_value[t] = (price[t] - SMA[t])/(2 * stdev[t])
```

Two other good features worth considering are momentum and volatility.

```
momentum[t] = (price[t]/price[t-N]) - 1
```

Volatility is just the stdev of daily returns.

Choosing Y

Your code should predict 5 day change in price. You need to build a new Y that reflects the 5 day change and aligns with the current date. Here's pseudo code for the calculation of Y

```
Y[t] = (price[t+5]/price[t]) - 1.0
```

If you select Y in this manner and use it for training, your learner will predict 5 day returns.

Contents of Report

- Your report should be no more than 2500 words. Your report should contain no more than 12 charts. Penalties will apply if you violate these constraints.
- Include the charts listed in the overview section above.
- Describe each of the indicators you have selected in enough detail that someone else could reproduce them in code.
- Describe your trading policy clearly.
- If you used any external code or ideas be sure to cite them in your code and in the report.
- Discussion of results. Did it work well? Why? What would you do differently?

Expectations

- In-sample sine and in-sample IBM backtests should both perform very well -- better than the manual policy you created for the last assignment.
- Out-of-sample sine backtest should perform nearly identically as the in-sample test.
- Out-of-sample IBM backtest should... (you should be able to complete this sentence).

What to turn in

Turn your project in via t-square.

- Your report as `report.pdf`
- All of your code, as necessary to run as `.py` files.
- Document how to run your code in `readme.txt`.
- No zip files please.

Extra credit up to 3%

Choose one or more of the following:

- Compare the performance of KNN and LinReg in this task. The instructor anticipates that LinReg might work well. If that turns out to be the case, how can that be? This is a non-linear task isn't it?
- Extend your code to create a "rolling" model that updates each day rolling forward.
- Extend your code to simultaneously forecast all the members of the S&P 500. Generate trades accordingly, and backtest the result.

Submit to the extra credit assignment on t-square. One single PDF file only, max 1000 words.

Rubric

- Are all 9 plots present and correct? -5 points for each missing plot.
  - Note: Correct in the sense that they properly display the information requested. The result may not be the desired one.
- Are comparative backtest results correct? (ML4T-240 in sample & out of sample, IBM in sample & out of sample) -10 points for each incorrect result.
- Indicators used: Are descriptions of factors used sufficiently clear that others could reproduce them? Up to -10 points for lack of clarity.
- Trading strategy: Is description sufficiently clear that others could reproduce it? Up to -10 points for lack of clarity.
- Is discussion of results concise, complete, correct? Up to -5 points for each of concise, complete, correct.

Required, Allowed & Prohibited

Required:

- Your project must be coded in Python 2.7.x.
- Your code must run on one of the university-provided computers (e.g. buffet02.cc.gatech.edu), or on one of the provided virtual images.
- Use only util.py to read data. If you want to read items other than adjusted close, modify util.py to do it, and submit your new version with your code.

Allowed:

- You can develop your code on your personal machine, but it must also run successfully on one of the university provided machines or virtual images.
- Your code may use standard Python libraries.
- You may use the NumPy, SciPy, matplotlib and Pandas libraries. Be sure you are using the correct versions.
- You may use scikit learn libraries (note that you don't need them because you just wrote your own!).
- You may reuse sections of code (up to 5 lines) that you collected from other students or the internet.
- Code provided by the instructor, or allowed by the instructor to be shared.
- A herring.

Prohibited:

- Any other method of reading data besides util.py
- Any libraries not listed in the "allowed" section above.
- Any code you did not write yourself (except for the 5 line rule in the "allowed" section).
