Summer 2019 Project 2: Optimize Something

From Quantitative Analysis Software Courses

Contents

Revisions

This assignment is subject to change up until 3 weeks prior to the due date. We do not anticipate changes; any changes will be logged in this section.

Overview

In this project you will use what you learned about optimizers to optimize a portfolio. That means that you will find how much of a portfolio's funds should be allocated to each stock so as to optimize it's performance. We can optimize for many different metrics. In this version of the assignment we will maximize Sharpe Ratio.

You can leverage the functions in the now deprecated (optional) assess portfolio project that assessed the value of a portfolio with a given set of allocations.

An older version of this project: MC1-Project-2-archive

Task

Implement a Python function named `optimize_portfolio()` in the file `optimization.py` that can find the optimal allocations for a given set of stocks. You should optimize for maximum Sharpe Ratio.

The function should accept as input a list of symbols as well as start and end dates and return a list of floats (as a one-dimensional numpy array) that represents the allocations to each of the equities. You can take advantage of routines developed in the optional assess portfolio project to compute daily portfolio value and statistics. You should cut-and-paste your code for the functions that did this into `optimization.py`.

You are given the following inputs for optimizing a portfolio:

- A date range to select the historical data to use (specified by a start and end date)
- Symbols for equities (e.g., GOOG, AAPL, GLD, XOM). Note: You should support any symbol in the data directory. Your code should support any number of assets >= 2. Don't hardcode 4.

Your goal is to find allocations to the symbols that optimize the criteria given above. Assume 252 trading days in a year and a risk free return of 0.0 per day. You should implement the following API EXACTLY, if you do not your submission will be penalized.

```
import datetime as dt
allocs, cr, adr, sddr, sr = \
    optimize_portfolio(sd=dt.datetime(2008,1,1), ed=dt.datetime(2009,1,1), \
    syms=['GOOG','AAPL','GLD','XOM'], gen_plot=False)
```

Where the returned output is:

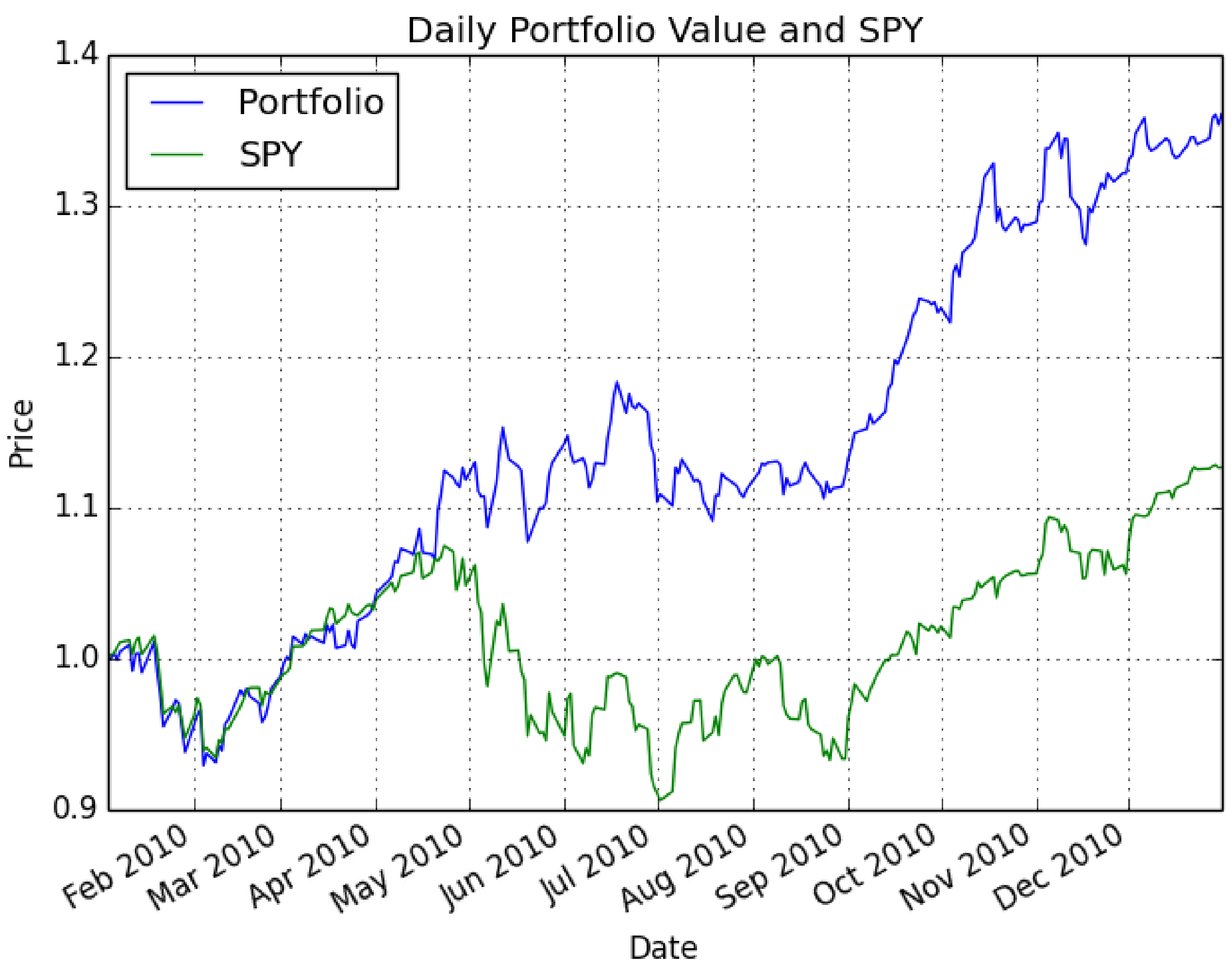
- allocs: A 1-d Numpy ndarray of allocations to the stocks. All the allocations must be between 0.0 and 1.0 and they must sum to 1.0.
- cr: Cumulative return
- adr: Average daily return
- sddr: Standard deviation of daily return
- sr: Sharpe ratio

The input parameters are:

- sd: A datetime object that represents the start date
- ed: A datetime object that represents the end date
- syms: A list of symbols that make up the portfolio (note that your code should support any symbol in the data directory)
- gen_plot: If True, optionally create a plot named `plot.png`. Autograder will always call your code with gen_plot = False.

ALL of your code should be present in the single file `optimization.py`

An example chart that you might create for assessing your optimizer.



Template

Instructions:

- Download the template code here: File:19spring optimize something.zip
- Implement the `optimize_portfolio()` function in `optimize_something/optimization.py`. The grading script for this project is **`grade_optimization.py`**.
- To execute your optimization code for debugging purposes, run **`PYTHONPATH=..:. python optimization.py`** from the `optimize_something/` directory.
- To run the grading script, follow the instructions given in ML4T Software Setup

To test your code, we will be calling `optimize_portfolio()` only.

Suggestions

- Refer to comments in the provided helper code for pointers regarding how to implement it. In order to specify bounds and constraints when using the `scipy.optmize` module, you'll need to use a special syntax explained here: http://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html

- For bounds, you simply need to pass in a sequence of 2-tuples `(<low>, <high>)`. Just remember that you need to supply as many tuples as the number of stocks in your portfolio.

- For constraints, it's a little tricky. You need to pass in a sequence of dicts (dictionaries), one dictionary per constraint. Each dictionary must specify the type of constraint (`'eq'` for equality, or `'ineq'` for inequality), and a function that *returns 0 only when the input satisfies the constraint* (this is the same input that is supplied to your evaluation function). E.g. to constrain the sum of all values in the input array to be less than 50, you could pass in the following (lambdas are just anonymous functions defined on-the-spot):

```
constraints = ({ 'type': 'ineq', 'fun': lambda inputs: 50.0 - np.sum(inputs) })
```

- Use a uniform allocation of 1/n to each of the n assets as your initial guess.

What to turn in

Be sure to follow these instructions exactly!

Via Canvas, submit as attachments (no zip files; refer to schedule for deadline):

- Your code as `optimization.py` (only the function `optimize_portfolio()` will be tested). Make sure that all necessary code is in that file. Please submit the code (.py) into the assignment "Project 2: Optimize Something (Code)" only
- A report `report.pdf` that includes a chart comparing the optimal portfolio with SPY using the following parameters: Start Date: 2008-06-01, End Date: 2009-06-01, Symbols: ['IBM', 'X', 'GLD', 'JPM']. Please submit the report (.pdf) into the assignment "Project 2: Optimize Something (Report) only

Unlimited resubmissions are allowed up to the deadline for the project.

Rubric

Part 1: Chart is correct [20 points]

- -20 no chart or chart is total nonsense
- -10 chart wrong shape (incl. wrong time period, etc) xor -5 chart partly wrong shape (diverges from correct halfway through, etc)
- -10 chart not normalized xor -5 chart data scale substantially wrong (right shape, normalized to 1, but max/end values wrong, 3.5 instead of 2.5 etc)
- -10 missing required data series (didn't plot one of portfolio or SPY)
- -2 chart labels/text/legend unreadable (too small, labels overlap a lot, etc) or missing.
- Min score: 0.

Part 2: 10 test cases [80 points]
We will test your code against 10 cases (8 points per case). Each case will be deemed "correct" if:

- sum(allocations) = 1.0 +- 0.02 (2 points)
- Each allocation is between 0 and 1.0 +- 0.02 (negative allocations are allowed if they are very small) (2 points)
- Standard deviation of daily return of the allocated portfolio is within 5% of reference solution or lower (4 points)
    - Deprecated: Each allocation matches reference solution +- 0.10 (4 points)

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. buffet01.cc.gatech.edu).
- Use the code for reading in historical data provided in util.py. Only use the API methods provided here to read in stock data. Do NOT modify this file. For grading, we will use our own unmodified version.
- Your code must run in less than 5 seconds on one of the university-provided computers.

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 (except os).
- You may use the NumPy, SciPy, matplotlib and Pandas libraries. Be sure you are using the correct versions.
- Code provided by the instructor, or allowed by the instructor to be shared.

Prohibited:

- Any use of global variables.
- Any libraries not listed in the "allowed" section above.
- Calls to code in any other local files besides util.py
- Use of any code other than util.py to read in data.
- Use of Python's os module.
- Any code you did not write yourself (except for the 5 line rule in the "allowed" section).
- Camels and other dromedaries.