CS7646 ML4T
Machine Learning
for Trading

PROJECT 2: OPTIMIZE SOMETHING

DUE DATE

09/06/2020 11:59PM Anywhere on Earth time

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

This assigment counts towards 3% of your overall grade.

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 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. Note: assess_portfolio is written in Python 2 (as it is deprecated) and will require refactoring to Python 3 to be useable. This is given only as a potential tool to use while completing the project.

TEMPLATE

Instructions:

- Download the template code here: File:Optimize_Something_2020Fall.zip
- Implement the `optimize_portfolio()` function in `optimize_something/optimization.py`.

- To execute your optimization code for debugging purposes, run **`PYTHONPATH=..:. python optimization.py`** from the `optimize_something/` directory.
- The grading script for this project is **`grade_optimization.py`**.
  - To run the grading script, follow the instructions given in ML4T Software Setup
  - NOTE: Passing **grade_optimization.py** does not gurantee passing the more stringent private grader. It is there are a starting point for you to use. Review the grader and the auto-grader rubric section to see what is/isn't tested. It is advised to test with multiple parameters to ensure your implementation is robust.

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

TASKS

Implement optimize_portfolio()

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 by using cut-and-paste for the code for the necessary functions into optimization.py. Otherwise, you will need to create the functions within 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 as the number of assets.

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

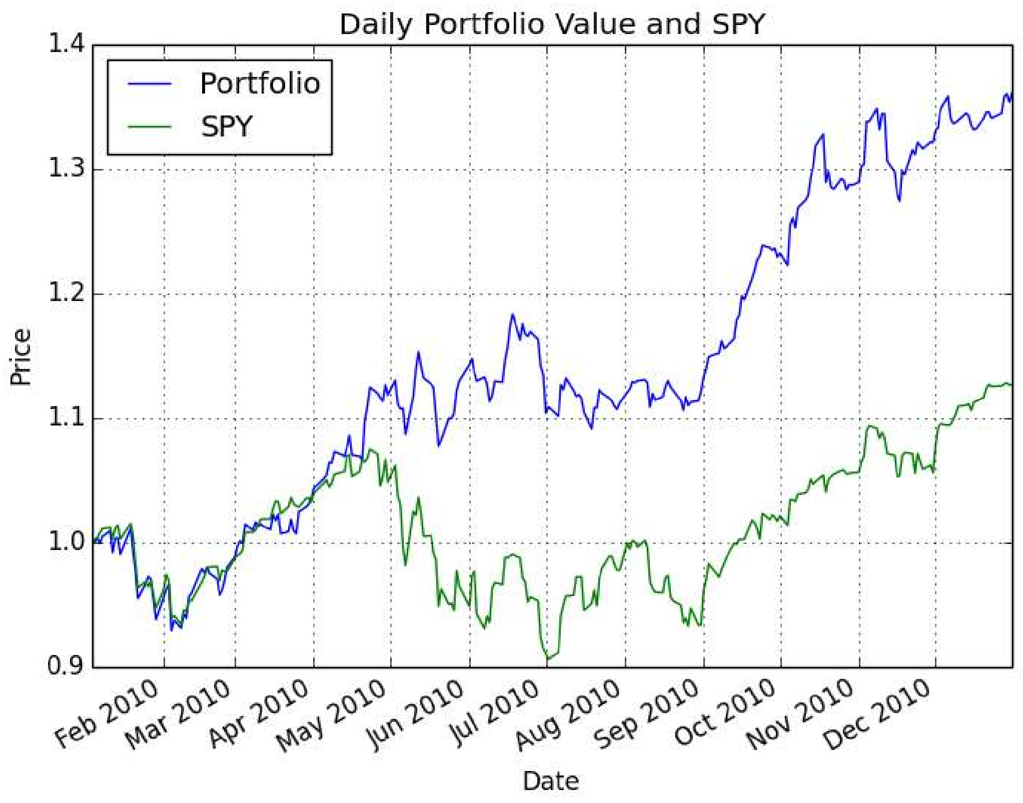
The input parameters are:

- 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`

Chart Example

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



Suggestions

- Refer to comments in the provided helper code for pointers on implementation. 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

Canvas:

Submit the following files (only) via Canvas before the deadline:

- Project 2: Optimize Something (Report)
  - Your report as `report.pdf` that includes a single 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’].

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

Gradescope:

- (SUBMISSION) Project 2: Optimize Something
  - Your code as `optimization.py` (only the function `optimize_portfolio()` will be tested). Make sure that all necessary code is in that file.

Do not submit any other files.

You are only allowed 3 submissions to **(SUBMISSION) Project 2: Optimize Something** but unlimited resubmissions are allowed on **(TESTING) Project 2: Optimize Something.**

Note that Gradescope does **not** grade your assignment live; instead, it pre-validates that it will run against our batch autograder that we will run after the deadline. There will be **no** credit given for coding assignments that do not pass this pre-validation.

Refer to the Gradescope Instructions for more information.

RUBRIC

Report [20 points]

Each chart line (SPY/Portfolio is worth 10 points each)

- -20 no chart or chart is total nonsense

- -10 chart wrong shape (incl. wrong time period, etc) or -5 chart partly wrong shape (chart begins correctly but midway or after is incorrect, etc)
- -10 chart not normalized or -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.

Code

- -10 Does the code generate appropriate charts written to png files? **DO NOT use plt.show() and/or manually save your charts. The charts should be created and saved using Python code.**

Auto-Grader [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)

There is no partial credit for the per-test-case points breakdown above if outside the threshold. If you are within the threshold you get the point allocation, outside the threshold you get 0.

REQUIRED, ALLOWED & PROHIBITED

Required:

- Your project must be coded in Python 3.6.x.
- Your code must be submitted to Gradescope in the appropriate Gradescope assignment.
- 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 per test case on one of the university-provided computers.

- Reference any code used in the “Allowed” section in your code. At minimum it should have the link/filename/video name of where it came from.

Allowed:

- You can develop your code on your personal machine, but it must also run successfully Gradescope.
- 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 use of plot.show()
- Absolute import statements of the **current** project folder such as `from optimize_something import XXXX` or `import optimize_something.XXXX`
- Extra directories (manually or code created)
- Extra files not listed in “WHAT TO TURN IN”
- Any code you did not write yourself (except for provided/allowed by the instructor).
- Camels and other dromedaries.