CPSC 537
Project Proposal

Chris Harshaw

November 27, 2016

The following is a brief outline of our CPCS 437/537 final project. We present the project description and reflect on the main requirements of our code.

1 Project Description

Our project is to develop an paper-trading app called "Portfolio Playground". *Paper-trading* is a term that refers to practicing stock investments without actually investing real money. Paper-trading is a beneficial learning experience for novice traders, as it builds important intuition. Our app will allow for intrepid traders to gain the valuable experience of paper-trading by providing methods to analyze, compare, and recommend stock portfolios. In the remainder of this section, we describe our proposed methods.

1.1 Portfolio Analysis

We consider two metrics for evaluating stock portfolios. The first is *total stock return* (TSR), a commonly-used performance metric. For a portfolio A consisting of stocks $A = \{s_1 \dots s_N\}$, the total stock return for the period t_0 to t_f is defined as

$$TSR_{t_0}^{t_f} = \sum_{i=1}^{N} x_i \frac{(P_i^{t_f} - P_i^{t_0}) + D_i}{P_i^{t_0}}$$

where x_i is the number of shares of stock s_i, D_i is the dividends for stock s_i, $P_i^{t_0}$ is the price of a single share of stock s_i at time t_0 and $P_i^{t_f}$ is the price of a single share of stock s_i at time t_f. The total stock return is the percentage of return for a given portfolio. In some sense, total stock return measures the success of a portfolio from time t_0 to t_f. The next performance metric is *stock diversity* (SD). This is not a commonly used metric, but rather one that we define. We claim it captures the diversity of a stock, which is important for novice traders. For a portfolio A consisting of stocks $A = \{s_1 \dots s_N\}$, the stock diversity for the period t_0 to t_f is defined as

$$SD_{t_0}^{t_f} = \frac{1}{N} \sum_{i<j} Cov(P^i, P^j)$$

We would like to analyze stock portfolios using these metrics.

1.2 Portfolio Comparison

Now that we have described how to analyze a single portfolio, we describe how to compare many portfolios against each other. The first comparison method is to plot the running TSR over time. Similarly, we can plot the running PD over time. Doing this for historical data will display a comparison of what actually happened. We can also compare the predicted performance in the following way: first, we fit our data to a *Autoregressive Integrated Moving Average* (ARIMA) model, a linear model commonly used in stock forecasting. Next, we run simulations and calculations on our model to obtain the expected TSR and SD for a given forecast range. We can combine historical and projected values of TSR and SD to compare portfolios.

1.3 Portfolio Recommendation

Our app will also support several portfolio recommendation algorithms. The first portfolio recommendation algorithm, `rand_portfolio` is simply a random selection of N stocks[1]. The next several recommendation algorithms rely on maximizing TSR under budget constraints, which we formulate as a linear-integer program. For all stocks $s_1, \dots s_K$, let x_i be the number of shares of stock s_i. Then we can maximize TSR over the range t_0 to t_f by solving the following linear-integer program:

$$\begin{array}{lllr}
\text{maximize} & \displaystyle\sum_{i=1}^{K} x_i \frac{(P_i^{t_f} - P_i^{t_0}) + D_i}{P_i^{t_0}} & & \\
\text{subject to} & x_i \geq 0 & i = 1 \cdots K & \text{(only nonnegative number of shares)} \\
& \displaystyle\sum_{i=1}^{K} x_j P_i^{t_0} \leq B & & \text{(total budget)} \\
& P_i^{t_0} x_i \leq M & i = 1 \cdots K & \text{(max investment per stock)} \\
& \|x\|_0 \leq N & & \text{(maximum number of stocks in portfolio)} \\
& x_i \in \mathbb{Z} & i = 1 \cdots K & \text{(integer number of stocks)}
\end{array}$$

Solving the linear-integer program above yields a portfolio with a maximum TSR under given budget constraints. Budget constraints may be added or removed for more realistic portfolio recommendations and comparisons, at the cost of higher computation time.

The second portfolio recommendation algorithm, `max_TSR_portfolio`, recommends the historically optimal stock from observed data by solving the optimization problem on historical data. The third portfolio recommendation algorithm, `max_expected_TSR_portfolio`, recommends the portfolio that maximizes expected TSR for a given forecast range by fitting an ARIMA model and solving the optimization problem on predicted data. The fourth portfolio recommendation algorithm, `expected_diverse_portfolio` is a greedy algorithm with inputs K, σ that works in the following way: (1) fit an ARIMA model to the stock pricings. (2) set the portfolio to $A = \emptyset$ (3) Let S be the set of stocks with covariance less than σ for all $s_i \in A$ (4) update A with the $s \in S$ with maximum expected TSR. (5) repeat steps 3 and 4 until $|A| = K$. Intuitively, this algorithm creates a diverse portfolio by adding the best stock that is sufficiently different from all current stocks.

2 Data Acquisition and Storage

Our database has two main types of data to store —the stock price data and the user portfolios. First, the database should contain stock prices for each stock in the market as well as dividends for each stock. The time-scale for the stock prices is an important design decision - it probably suffices to store daily records. Stock prices will be obtained by an API. Next, the user information such as user name and password must be stored. Finally, we should store the portfolio information for each user as well. This includes the stocks in the portfolio, the number of shares of each stock, and the purchase date.

There are several types of queries that our database will have to handle. The main query is that of selecting the prices for a set of stocks A from time t_0 to time t_f. It is important that this query is fast, as we will use it frequently as a sub-query. The structure of the database is an important design decision that will influence the performance of our app.

3 User Interface and Visualizations

The user interface is an important design decision that will have a large impact on user experience. I have absolutely zero input here.

There are several types of visualizations that are important for the analytic portion of the app. the first type of plots are TSR plots. When considering a single portfolio, it would be interesting to see a break down

[1] this random selection is an optimal recommendation algorithm under the Efficient Market Hypothesis. As such, this option allows us to 'test' the efficient market hypothesis, which I believe to be a very silly hypothesis.

of the TSR of each individual stock over time, in addition to the TSR of the entire portfolio. When two portfolios are being compared, it is probably sufficient to look at the TSR of the portfolios being compared over time. To visualize SD, we can plot SD over time. To get a more refined view, we can also plot a cluster-ed heatmap of the covariance between prices of stocks in the portfolio. These plots also allow for comparisons of SD between portfolios.

4 Required Computations

There are several computations that we need to have on the backend. They are listed here in no particular order.

- Fitting an ARMIA model to stock pricings
- Solving maximum TSR of a portfolio A over period t_0, t_f
- Recommender algorithms 1 thru 4
- Clustering for heat-map

We can take advantage of existing software for a lot of these tasks.