Stock Portfolio Selection Using Two-tiered Lazy Updates

Alexander Cook

Submitted under the supervision of Dr. Arindam Banerjee to the University Honors Program at the University of Minnesota-Twin Cities in partial fulfillment of the requirements for the degree of Bachelor Science, summa cum laude in Computer Science.

May 9, 2014
Acknowledgements

This thesis could not have been completed without the help and support of Dr. Banerjee. He is one of the busiest people I know, yet he regularly found time to help me one on one. After every meeting, he left me excited and energized about our work and its potential for solving real-world problems. Thank you to Nicholas Johnson for letting me raid his office hours to discuss my results and brainstorm ideas for improving my algorithm. Thank you to Dr. Steve Jensen for converting me to a Computer Science major, and then connecting me with Dr. Banerjee. Finally, my undergraduate career would not have been nearly as successful without the outstanding advising and support from Jeanne Anderegg—I would not be graduating with honors if it were not for you. Thank you all for your help and guidance.
Abstract

People make and lose vast sums of money every day on stock exchanges around the world. This research focused on developing a computer algorithm to build profitable portfolios, while taking into account transaction costs associated with trading stocks. The theory behind our algorithm is based on a subset of Machine Learning called Online Learning (Cesa-Bianchi and Lugosi 2006). Online Learning makes updated decisions as new information is provided. For our case, a new decision is made each day on what stocks to buy/sell based on transaction costs and the previous day’s stock performance. The Lazy Update part of our algorithm seeks to minimize the quantity of trading, since this leads to transaction costs being incurred. Our algorithm builds on prior work and dynamically learns which sectors to invest in and takes into account risk, which has not been considered before in the literature. Our Online Lazy Updates algorithm runs at a low level on choosing stocks within a sector, and at a high level on choosing the best sectors to invest in. We successfully establish our ability to be profitable with transaction costs on real-world data.

1 Introduction and Motivation

Developing an algorithm that consistently makes large sums of money is the dream of many computer scientists. With the benefit of hindsight, finding the best portfolio is a combinatorial problem on the order of the number of stocks exponentiated to the number of time steps. A limit can be placed on the number of trades, to reduce the state space. Observing that the batch case of Portfolio Selection
has optimal substructure can further reduce the problem, allowing for the use of dynamic programming methods.

Solving the batch case offers little practical use beyond providing an upper bound to predictive algorithms. Predictive algorithms—especially those that take transaction costs into account—are needed in order to solve the online portfolio selection problem in the real world.

There have been promising results achieved by the Online Lazy Updates (OLU) algorithm (Das et al. 2013) while taking into account transaction costs. Despite the strong performance of the OLU, professionals in the finance industry will not touch algorithms that do not take portfolio risk into account. In this paper, we introduce a two-tiered online portfolio selection algorithm with transaction costs that takes into account sector group structure and sector risk. In the lower tier, the algorithm runs OLU on the stocks within each sector. Sector performance under the OLU is then fed into the upper tier, which runs a sector version of the OLU algorithm (SOLU).

Our batch experiments were conducted on two real world datasets: S&P 500 dataset with 263 stocks (1990-2010) and NYSE dataset with 36 stocks (1962-1984). The online experiments were conducted on the S&P500 dataset. Our experiments show that our online portfolios are able to make money under the restriction of transaction costs. At the market level we are able to track the performance of the best sector, even with transaction costs.
2 Batch Case Portfolio Selection

The batch case of portfolio selection requires perfect knowledge of the daily stock prices for each stock over the time period of investment. The transaction cost must be known, either as a percentage or fixed charge. This level of information is obviously unrealistic under normal circumstances, but this work shows the upper bound of wealth accumulation in the stock market. The naïve, combinatorial solution to this problem requires enumerating all possible portfolios (on the order of the number of stocks exponentiated to the number of time steps) and selecting the one with the highest final value. A dynamic programming solution can take advantage of the optimal substructure of portfolio selection to greatly reduce the required computation.

2.1 Dynamic Programming Recurrence

Throughout this paper, we use $x_t$ to denote a vector of stock price relatives on day $t$. A price relative is the multiplicative change in stock price for that day. $\gamma$ is the transaction cost as a percentage of a trade and $c$ is the initial investment into a portfolio. The optimal wealth at time $t$, holding stock $i$, after $k$ trades, $w^*_t(i, k)$, is given by the following recurrence:

$$w^*_t(i, k) = \begin{cases} 
  c \cdot (1 - \gamma) \cdot x_t(i) & \text{if} \quad t = 0 \\
  \max \left\{ w^*_{t-1}(i, k) \cdot x_t(i), \max_{1 \leq l < S} \left\{ w^*_{t-1}(l, k-1) \cdot (1 - \gamma) \cdot x_t(i) \right\} \right\} & \text{else} 
\end{cases}$$

(1)

To solve the final optimal solution, $w^*_t(i, k)$ must be computed for all values of $k$ within the limit of trades $K$, $k \in K$, at every time-step, $t \in T$, over all stocks, $i \in S$. The upper bound of $K$ is $T$, which allows for a trade every day. At any given time,
only one stock is held in a portfolio. This is because the best portfolio over a set time is always the best performing stock over that time. The optimal portfolio can be found by tracing back the stocks that maximized (1) from the final optimal wealth. This results in a space complexity of $O(T^* S^* K)$, or $O(T^* K)$ if you make the reasonable assumption that the number of stocks $S$, is a constant. The time complexity is given by $O(T^* S^2 K)$, or $O(T^* K)$ under the constant number of stocks constraint.

2.2 Results

As would be expected, the optimal solution to the batch case of portfolio selection results in the accumulation of a vast sum of money.

Table 1: Wealth accumulation, $w^*$, for varying trade limit, $K$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initial Investment</strong></td>
<td>$1</td>
<td>$1</td>
</tr>
<tr>
<td><strong>Transaction Cost</strong></td>
<td>1%</td>
<td>1%</td>
</tr>
<tr>
<td>$w^*$ for $K=10$</td>
<td>$19.2$ billion</td>
<td>$35.3$ million</td>
</tr>
<tr>
<td>$w^*$ for $K=20$</td>
<td>$463$ trillion</td>
<td>$18.1$ billion</td>
</tr>
<tr>
<td>$w^*$ for $K=100$</td>
<td>$4.36 \times 10^{33}$</td>
<td>$2.96 \times 10^{21}$</td>
</tr>
</tbody>
</table>

Any reasonable transaction cost plays almost no effect on the absurdly high wealth gains because the benefits of the carefully selected trades strongly outweigh the cost of trading. For a transaction cost that is set very high (ex. 50%), the dynamic program may not use up all $K$ available trades.

Table 2: Ideal 10 trades to earn $19.2$ billion on S&P500

<table>
<thead>
<tr>
<th>Trade</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stock Symbol</td>
<td>DELL</td>
<td>BBY</td>
<td>EMC</td>
<td>DELL</td>
<td>SCHW</td>
<td>ORCL</td>
<td>THC</td>
<td>TSO</td>
<td>SWN</td>
<td>F</td>
</tr>
</tbody>
</table>
3 Online Portfolio Selection

The batch case solution to portfolio selection is obviously not useful for making money on the stock market. Online solutions are necessary as a way to use data before it loses its value. Understanding what was trending on Twitter one month ago is not nearly as valuable as knowing what is trending now. This means that we require scalable solutions that can handle data at a velocity of 500 million tweets per day. In online portfolio selection, we try to make predictions about how stocks will perform before the information is available to us.

In the batch case, we set a limit to the number of trades, $k$. In the online case, we do not explicitly limit $k$, but we seek to minimize trading by penalizing transactions. The online case also makes use of group information as a way to incorporate side information.

3.1 Related Work

This paper directly extends work on the OLU algorithm proposed by (Das et al. 2013). The OLU paper was one of the first to address the need to consider transaction costs brought up by (Helmbold et al. 1998; Cover and Ordentlich 1996; Cesa-Bianchi and Lugosi 2006; Agarwal et al. 2006). Other than (Das et al. 2013), (Blum and Kalai 1997) are the only ones to extend the analysis of (Cover 1991) to include proportional transaction costs in the algorithm design. They were able to show that the performance guarantee of the Universal Portfolio holds for a time and then degrades as transaction costs become too high. The OLU algorithm has stronger empirical results than (Blum and Kalai 1997), converges faster in practice,
and incorporates transaction cost directly into the convex optimization step of portfolio calculation as a way to reduce the amount of trading.

The use of side information has also been explored by (Helmbold et al. 1998; Cover and Ordentlich 1996). As looked at by (Helmbold et al. 1998), they are able to distinguish between useful and useless side information without much loss in performance for their algorithm.

3.2 OLU Algorithm

We formulate a two-tiered online lazy portfolio selection strategy that allows us to control the amount of trading within groups and between groups. First the stocks are assigned groups (one stock can be assigned to multiple groups). We used sector information to form the groups. Unlike the incorporation of side-information in (Helmbold et al. 1998) (which was robust to useless side information), for this problem formulation, the quality of grouping plays a key role in the benefit of a two-tiered approach. Poor grouping will lead to poor overall performance.

Diagram 1: Diagram of the Two-Tiered OLU
Sector Level:

At the sector level, we represent a sector portfolio at time $t$ as a vector $q_t$ of weights that correspond to the distribution of wealth between stocks in that sector. Then we use the (Das et al. 2013) OLU formulation where $x_t$ is the vector of stock price relatives on day $t$, $\beta$ is the weight on the log-loss of the portfolio, and $\tau$ controls the amount of trading that takes place:

$$ q_{t+1} = \arg\min_{q \in \Delta_n} -\beta \log(q^T x_t) + \tau \| q - q_t \|_1 + \frac{1}{2} \| q - q_t \|_2^2 \quad (2) $$

The purpose of the first term is to maximize the logarithmic wealth if the known price relatives, $x_t$ were to be repeated at time $t+1$. The second term is an $\ell_1$ penalty that weights the amount of trading that will occur. High values of $\alpha$ lead to small amounts of transactions, ideally used under the restriction of high transaction cost. Low values of $\alpha$ allow the portfolio to change frequently.

Market Level:

At the market level, we represent a portfolio at time $t$ as a vector $p_t$ of weights that correspond to the distribution of wealth between different sectors. Then we use a variation of the OLU formulation where $z_t$ is the vector of modified sector price relatives on day $t$, $\eta$ is the weight on the log-loss of the market level portfolio, and $\alpha$ controls the amount of trading that takes place between sectors:

$$ p_{t+1} = \arg\min_{p \in \Delta_n} -\eta \log(p^T z_t) + \alpha \| p - p_t \|_1 + \frac{1}{2} \| p - p_t \|_2^2 \quad (3) $$

Both (2) and (3) are solved using an ADMM (Alternating Direction Method of Multipliers (Boyd et al. 2011)) based primal-dual algorithm as in (Das et al. 2013).
Modified Sector Price Relatives:

The vector $z_t$ at the market level of the OLU is based on the performance of the OLU algorithm on each sector. Day $t$’s price relative, $\delta_t$ for a given sector is the multiplicative change in wealth of the sector portfolio on that day. We further modify the price relative to take into account risk, where we treat risk as a function of the sector’s wealth gain for the last $d$ days, over the standard deviation of the sector price relatives over that time period:

$$\delta_t' = \frac{\prod_{i=d+1}^{t} \delta_i}{\max\{c, \sigma(\delta_{t-d+1})\}} \tag{4}$$

The selection of $c$ determines a risk tolerance. For large values of $c$, the standard deviation ($\sigma(\delta_{t-d+1}) \sim 0.01-0.04$ over 30 days) will always be less than $c$ and we will only take into account the wealth gain over the last $d$ days. For very small values of $c$, the risk term will always supersede $c$. The standard deviation and variance are commonly used as a simple measure of the volatility of stocks. Stocks that have remained steady in the past are less likely to crash (albeit less likely to see huge gains), therefore reducing the level of risk. Each $\delta'_t$ is normalized through dividing by the mean of $\delta'_t$ over all sectors. This modified price relative ranges mostly from $\sim 0.90-1.10$, compared to the tighter bound of $\sim 0.97-1.03$ for the sector price relatives.
3.3 Results

Data Set

The experiments were conducted on the Standard & Poor’s (S&P 500) (Das and Banerjee 2011) dataset. The dataset consists of the 263 stocks that were present in the S&P500 index from 1990 to 2010. This period captures the dot-com bubble from 1997 to 2000, the bubble burst in March 2000, and the recent housing bubble burst of 2007. This dataset does not contain stocks that went bankrupt during this time, fell from the S&P500, or entered after 1990.

Methodology

In all the experiments, we start with an initial investment of $1 into a portfolio that is uniformly distributed across the stocks within each sector, and across the sectors. Starting at $t=d+1$, we update the sector portfolios and the market portfolio by solving (2) and (3) respectively. One of the challenges of running experiments for the two-tiered OLU is the number of parameters. The focus of this work was on the market level, so the parameters for each sector OLU were kept constant and the same across sectors. The parameters we varied were $\eta$: weight on the logarithmic gain in wealth across all sectors, $\alpha$: the weight on the $\ell_1$ trading penalty, $\gamma$: fixed percentage between-sector transaction cost, and $c$: the level of risk tolerance. Results shown are for a modified price relative where the number of days, $d=30$. 


Effect of $\alpha$ and $\eta$ on wealth

**Figure 1**: Effect of $\alpha$ and $\eta$ when there are no transaction costs from two angles. A balance between $\alpha$ and $\eta$ lead to the highest wealth gain.
Figure 2: Effect of $\alpha$ and $\eta$ with 0.1% transaction cost from two angles. Highest wealth gains are seen when $\alpha$ dominates $\eta$. 
Under no transaction cost, a balance between the weight on the wealth gain ($\eta$) and the weight on the transaction term ($\alpha$) leads to the strongest results. If $\eta$ is too large, the portfolio changes too quickly and “forgets” about the trend that it is trying to catch. If $\alpha$ is too large, the portfolio doesn't change at all and misses opportunities for wealth gain.

Surprisingly, there does not seem to be a balance between the wealth gain ($\eta$) and the weight on the transaction term ($\alpha$) that leads to the strongest results. Rather, strongest results occur when $\eta$ is $\sim 1.0$ and $\alpha$ is any number greater than 0.5. This would seem to indicate a threshold where there is no longer any trading occurring (since the weight on the transaction penalty is dominating), but that is not the case. What ends up occurring is that most day, the algorithm does not trade at all, but when it does, it moves the entire weight of the portfolio into one new sector. We experimented extensively on this behavior because it seems to be a contradiction: $\alpha$ dominating (the lazy term) should lead to no trades at all.
Controlling Risk

Figure 3: Plot of the wealth gain over time for varying levels of risk. As the tolerance for risk, $c$, increases, the achieved wealth also increases.

As expected, the more risk you are willing to take on, the greater wealth you are able to achieve over a long period of time. However, low-risk portfolios were only marginally better at avoiding recession losses than high-risk portfolios. Low $c$ values ($c=.01,.015$) remain almost flat during the dot-com bubble. During the burst of the housing bubble, the low-risk portfolios still dropped $\sim50\%$, while high-risk portfolios dropped $\sim65\%$. We expected that our model for risk would result in lower wealth gains during upswings in the economy (as we observed), but we were also expecting that low-risk portfolios would remain steadier during recession (not the case in this dataset).
Tracking the best sector

Figure 4: For a 0.1% transaction cost we are able to reasonably track the best OLU sector (the tech sector).

A reasonable way to measure how well the market-level OLU is performing is to compare it to the performance of the OLU on each sector. The tech sector vastly outperforms any other sector and we are unable to surpass it with a transaction cost. However, for transactions costs of 0.1% and 1.0%, we are able to outperform all other sectors. The ability to track the best sector without the benefit of hindsight or prior information is impressive, considering the simplicity of objective functions (2) and (3). These results are not nearly as strong as those in (Das et al. 2013), but this is to be expected since there was no fine-tuning of the OLU on each individual sector and the grouping of stocks by sector is a simple choice of grouping.
4 Conclusion and Future Work

In this paper, we have examined the portfolio selection problem in both the batch case and the online case. We were able to solve the batch case for a restricted number of trades using dynamic programming and we showed that only a few trades can result in massive wealth gain with the benefit of hindsight. We have built on the OLU algorithm (Das et al. 2013) to include group information and risk by creating two tiers of OLU computation. Our experiments show that the two-tiered OLU is able to remain competitive with the best OLU sector, even under the constraint of a small transaction cost.

In the future, we wish to explore learning of the group structure that could lead to more beneficial groupings than the sectors used in this paper. We would like to see a more effective and practical implementation of risk, as this is a major concern before any portfolio selection algorithm can be used in practice. Another major concern is the selection of model parameters. Future work on parameter estimation and optimization would be another key step for making the OLU more practical for real world application. Finally, we envision the incorporation of group information as being relevant to problems in other domains such as social media analytics.
References


