Dynamic Learning Rate Adjustment Algorithm

Brian Bullins, Sergiy Popovych, Hansen Zhang

Abstract

Developing an investment portfolio for the stock market that will yield positive returns is the primary goal of investors worldwide. A variety of models and algorithms have been developed to decide upon a distribution which maximizes gains in the market, a few examples being the Geometric Brownian Motion model and the universal portfolio. One particular algorithm, known as online gradient descent, can be used for portfolio management with reasonable success. Its performance depends heavily, however, on the choice of the learning rate η, and it is difficult to know *a priori* which values will yield the best results. In our project, we explored various means of adjusting η based on different attributes of the stock market in an attempt to determine a more systematic approach to dynamically learn profitable values of η. One especially promising algorithm involved an implementation of multiplicative weights, where each expert is represented by a gradient descent algorithm with a unique value for η. To determine the profitability of our algorithms, we tested them on a variety of long-term daily-resolution stock data sets from many different markets.

1. Introduction

An investor's main objective in the stock market is to determine how to place his money among stocks so as to reap the greatest reward while at the same time maintain a low amount of risk. The field of finance theory focuses heavily on developing investment models that will allow for maximizing wealth gains, and some online algorithms have been fit to the problem of portfolio management to further this objective. One such algorithm is the multiplicative weights method [1], whereby there is a set of experts among which we must choose one randomly for each round of our process, and each expert has a weight which increases or decreases depending on what cost was incurred by the expert in the previous rounds. The online gradient descent algorithm [2] embodies a similar notion of making a future prediction based on the previous observation. Instead of maintaining a set of experts, however, the online gradient descent algorithm involves moving in the direction of the gradient to hopefully further minimize the objective function. It is important to note that, in the case of portfolio management, because we are trying to maximize profit, our algorithm will need to make movements in the positive direction of the gradient, which may be known as gradient ascent since the goal in this version to find a maximum. One algorithm may easily be transformed into the other, however, by simply inverting the signs, so from this point on we shall use the term gradient descent, with the hope that the reader understands from context what is meant.

One of the critical components of the gradient descent algorithm, as we will see, is the choice of the learning rate parameter, η. In the context of the stock market, we make a choice at each round of a distribution of our current wealth among the stocks available and invest this money at the days opening price, and our payoff is the proportion of the amount of money in hand at the end of the day, with respect to the amount had at the beginning of the day. The distribution choice for each new day is determined as a function of the distribution of the previous day, along with an update in the positive direction of the gradient, times the η parameter. When η is 0, the distribution is not updated at all, so if the initial distribution is uniform, then it remains uniform for the entire process. For values of η greater than 0, it is generally the case that smaller values of η result in placing a slightly increased proportion of wealth in stocks that performed well on the previous day (and, by extension, removing wealth from poorer performing stocks), while larger values of η result in a much larger proportion of wealth being places in just a few top performing stocks of the previous day.

It was precisely this parameter η that piqued our interest during the implementation of the gradient descent algorithm in homework 4. Through trial and error, we found that some values of η resulted in disproportionately higher profitability. Curious to determine the reasons behind this occurrence, we tried to vary η based on a variety of different properties of the stock market, including volatility and general market growth. Through these explorations we managed to develop a modified version of the multiplicative weights algorithm, which treats each expert as its own gradient descent algorithm, but with a different value of η assigned to each expert. Our paper is outlined as follows: in section 2 we discuss some of the related work that influenced and guided our project, as well as the approaches we ultimately decided to take. Section 3 deals with the importance of choosing η, and its influence on the ultimately profitability of the gradient descent algorithm as applied to real-life data. In section 4, we cover the various heuristics we tried to apply to the market to dynamically determine η. In sections 5 and 6, we discuss the algorithms we created that yielded much more impressive performances than any of the heuristics attempted in the previous section. Sections 7 and 8 provide details about the data sets which we tested our algorithms against, as well as the wealth gains that came as a result.

2. Related Work

Elad et al. [3] presented an efficient algorithm for regret minimization with exp-concave loss functions, and improved upon the state-of-the-art algorithm in that aspect. Its implications in the Geometric Brownian Motion model section analyzed the relationship between variance and trading frequency inspired us on some level. This paper also assumed no transaction costs, which is an assumption we have made as part of our project. Kalai et al. [4] presented an efficient randomized approximation of the Universal algorithm. It has w.h.p. within $(1-\epsilon)$ times the performance of the Universal algorithm and runs in time polynomial in $\log(1/\eta)$, $1/\epsilon$, the number of days, and the number of stocks. This is important for practical concerns.

The multiplicative weights algorithm, as described in the survey by Arora et al. [1], works by assigning weights to a series of experts which are updated at each time step based on the cost function that is applied. At the end of each round, an expert is randomly chosen with probability that is proportional to its weight. For the online gradient descent algorithm [2], we move in the direction of the gradient in the hopes that our new choice from the decision set K (after projecting onto K) results in a smaller result from a given cost function. Interestingly, in a notable ICML paper from 2003 [5], Zinkevich showed that that some of the results that apply to general gradient descent can be naturally extended to the online case.

3. Importance of Eta

As part of homework 4, we were asked to implement and run the online gradient descent algorithm on a set of real S&P 500 stock data, consisting of 490 different stocks. After programming the main components of the algorithm, the only thing that is left to the person running the algorithm is to make a choice for the value of η. We discovered, after trying many different values of η, that the wealth gains for this stock data are much greater for higher values of η than for lower values.

Intrigued by this sensitivity to the choice of η, we decided to observe what would happen if we chose different values of η for smaller periods of time within the entire time period of the data set. After adjusting the individual values of η to be profitable for each month, we obtained a return on investment of more than 2000x. In addition, we found that these profitable values of η tended to vary wildly from month to month.

Overall, it tended to be large values of η that performed well for this original data set. Wondering if this would hold true for additional data sets, we made use of online resources and acquired 14 years of S&P 500 stock data. This data contains the open, close, high, and low prices of 402 stocks of the time period starting from January 3rd, 2000 and ending on December 31st, 2014, much larger than the one used for the homework. We applied the same online gradient descent algorithm from the homework on this new data set, and this time a large η value actually resulted in a loss of money. This observation suggested that a large η will not always yield profitable results.

Taken together, these results led us to consider the possibility of finding a correlation between the optimal η's of adjacent time periods that we might exploit through the decisions we make over long periods of time.

4. Heuristics Attempted

Our first approach to exploiting the algorithm's dependence on η involved generating a set of statistics about the state of the stock market. The volatility of the market can function as an indicator of the dispersion of prices, so we attempted to make use of this knowledge in determining how to adjust η. First, we tried increasing η as the market's volatility increased, and vice versa, with the rationale that if the volatility is increasing, it might be advantageous to adopt to the highly fluctuating market quickly. After lackluster performance, we tried the inverse procedure: when the volatility is high, decrease the value for η. We reasoned, in this scenario, that the more volatile the market, the more noise that exists making it more difficult to predict the trajectory of the market, leading the gradient descent algorithm to be less reliable. Unfortunately, our results were again underwhelming.

The next approach we took involved looking at the general market trends as exhibited by the stock prices. It is reasonable to imagine that if the market is growing as a whole, then there might be a few stocks that are growing especially quickly, and a larger value of η would allow for the algorithm to take advantage of this possibility. Thus, we tried increasing η as the market as a whole grew, but only to result in another mediocre performance. In order to be comprehensive in our tests, we tried the opposite whereby η decreases as the market begins to grow, though not surprisingly this also ended up as unprofitable.

We tried observing the distance between the gradient prediction of the stock change and the actual stock change, with the idea that the greater this distance, the smaller the value of η should be, as it would indicate an abrupt shift in the market. We also thought to try assigning an η to each stock which can vary on its own. This seemed reasonable, as then stocks behaving predictably and unpredictably have their own values of η that can be used to take advantage of these differences. Neither of these proposals, however, yielded much profitability. Overall, the mediocre results achieved through all of these trials led us to conclude that it is difficult to find properties of the stock market from which we can determine reasonable values of η.

5. Using Multiplicative Weights

After our underwhelming results mentioned in the previous section, we decided to try using the multiplicative weights algorithm to determine profitable values of η. Our model consists of a set of experts as part of the multiplicative weights algorithm, with each expert corresponding to a single gradient descent algorithm that has a unique assignment to its learning parameter η. Specifically, there are 6 experts, which take on these values of η: $\{0,\ 0.001,\ 0.1,\ 1,\ 10,\ 10000\}$. These values were chosen to cover a reasonable range of possible η's.

It is necessary for us to make some assumptions about the stock market, namely that there is an inherent structure or momentum to the data. Clearly, if prices can become arbitrarily large or small in a single day, we would have little hope of determining the flow of prices based on the previous data. It follows from this assumption that if an algorithm performs well for time period $t_i, \ldots, t_j$, we can reasonably expect it to perform well in a subsequent time period $t_{j+1}, \ldots, t_k$.

Another important aspect of our algorithm is that the lengths of the time periods are not so short as to be influenced by the randomness of the stock data. At the same time, if the periods are made too long, the model might fail to capture changes in the stock trends. Thus, the length of the time period chosen needs to balance these competing interests.

In order to take advantage of these assumptions, our multiplicative weight algorithm had two main parameters. One of these was the length of the time period after which we change the corresponding weights of the experts. The second parameter determines the number of changes after which the value of η is updated. To adopt to changes in the market, we decided to reset the weights after each new choice of η. If we did not make this adjustment, only long-term profitable values of η will be chosen, thus decreasing the chance of choosing an η which has performed well in a recent period of time.

We update the weights in the following way: we split the look-back period into equal sized chunks, and observe how each of the stocks performs over the course of an entire chunk. After the period of each chunk completes, we observe how all of the stocks perform relative to the highest performer. If a stock has a return that is at least 0.9 times that of the best stock, then its weight is doubled–otherwise, its weight is divided by 2. This worked quite well as a first attempt, but we still discovered a flaw in the reasoning behind it. Because of how we decided to update the weights based on the look-back period, we realized that the multiplicative manner of how the weights should be updated was instead dominated in an additive manner. This observation led us to update the algorithm to account for this issue.

6. Deterministic Version

After some initial success the multiplicative weights-based algorithm detailed in the previous section, we decide to develop an updated version of our algorithm. In our final, and most successful, version we deterministically choose the η that gives the best return over the previous N days, where N in the length of the look-back period. To make such a decision process possible, a separate money distribution is kept for each of the η's in the pool. However, after each new choice of η, all of the money distributions are reset to the one of the current η.

7. Data Sets

1. **S&P 500**

   S&P 500, or the Standard & Poors, is an American stock market index based on the market capitalizations of 500 large companies having common stock listed on the NYSE or NASDAQ. For this stock market index we used the data of 402 stocks across the span of 3773 days (14 years).

2. **ASX 200**

   The ASX 200 index is a stock market index of Australian stocks listed on the Australian Securities Exchange from Standard & Poor's. For this stock market index we used the data of 91 stocks across the span of 3800 days.

3. **FTSE 100**

   The FTSE 100, informally the "footsie", is a share index of the 100 companies listed on the London Stock Exchange with the highest market capitalization. For this stock market index we used the data of 68 stocks across the span of 3900 days.

4. **DOW JONES 30**

   The DOW JONES 30, or the Dow Jones Industrial Average, is a stock market index, and one of several indices created by Wall Street Journal editor and Dow Jones & Company co-founder Charles Dow. It is an index that shows how 30 large publicly owned companies based in the United States have traded during a standard trading session in the stock market. For this stock market index we used the data of 29 stocks across the span of 3773 days.

5. **NIFTY**

   The NIFTY, or the CNX Nifty, also called the Nifty 50, is National Stock Exchange of India's benchmark stock market index for Indian equity market. It is owned and managed by India Index Services and Products (IISL), which is a wholly owned subsidiary of the NSE Strategic Investment Corporation Limited. For this stock market index we used the data of 29 stocks across the span of 3000 days.

6. **NASDAQ 100**

   The NASDAQ 100 is a stock market index made up of 107 equity securities issued by 100 of the largest non-financial companies listed on the NASDAQ. For this stock market index we used the data of 75 stocks across the span of 3773 days.

7. **SSE**

   The SSE is the stock exchange of Shanghai. For this stock market index we used the data of 470 stocks across the span of 3800 days.

8. Results

Before applying our algorithm to the stock data of multiple markets, we first thought about what assignment of parameters of our algorithm would make logical sense. Using this as our guideline, we determined that it would not be reasonable to update η more often than once every 10 days. It also seems that if η is updated less often than once every 30 days, the model loses its flexibility to adapt to the market. For the look-back parameter, if it is shorter than 60 days, then we lose some information that would have otherwise been useful to provide a sense of the recent history of the stocks. At the same time, looking back for a period of more than a year tends to not pick up on the more recent trends that influence the trajectory of the prices. After some testing, our beliefs of reasonable choices of the update and look-back parameters were confirmed by the profitable results of applying our algorithm to the 14 years of S&P 500 stock data.

As can be seen from the figures below, our deterministic algorithm performs quite well on all of the data sets, with a truly remarkable performance in the Shanghai stock market. In figure 1, we see the performance of our algorithm on all of the data sets excluding Shanghai, with an update parameter set to 10, and a look-back parameter set to 100. It is easy to observe the relatively strong performance achieved by our algorithm on all of the data sets found in figure 1, with the Australia data set yielding the highest gains. Figure 2 exhibits the results of the algorithm behind applied to the same 5 data sets, but with a different assignment of parameters. It is worth noting the robustness of the algorithm, as even after adjusting the parameters, the returns on investment for all of our data sets remain reasonably strong (with the exception of Australia, which comes down from an incredibly high 301.5x return to a more reasonable, yet still impressive, 6.34x return). In fact, the average of roughly 8x return over approximately 14 years would indicate an average annual return of nearly 16%, which is a rather strong annual return.

It is obvious from our decision to use an additional figure that the performance of our algorithm on the Shanghai data is very different from the performance for the rest of the data sets. As we can see in figure 3, for large enough values of η, our algorithm results in more than a trillion times return on investment, which is absolutely phenomenal. Figure 4 shows a breakdown

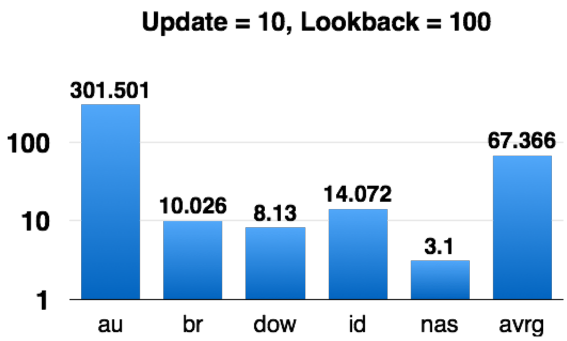

Figure 1: All markets, log scale

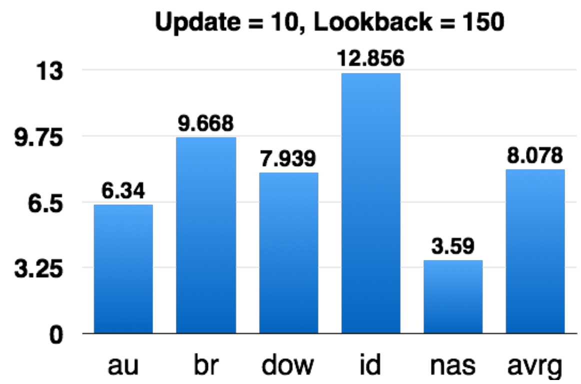

Figure 2: All markets

of the gains of our algorithm at intervals of 200 days, and we are unsure of the reason behind this astronomical performance. We believe that if it were due to a bug in our algorithm, then there would be similarly aberrant results for the other data sets. It is also possible that it is due to a bug in the stock data, but we are unsure of the best way to determine if the data itself is corrupted. One reason to suspect that it is not simply a bug in the data due to what can be seen in figure 4. There we seen a consistent exponentiation of the data from day 1800 onwards, which is what one would expect for general trends in the stock data. For values of η less than 100, however, this phenomenon is not observed.

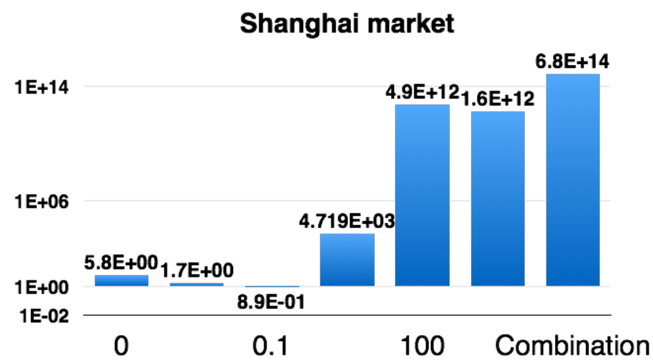

Figure 3: Shanghai only, log scale

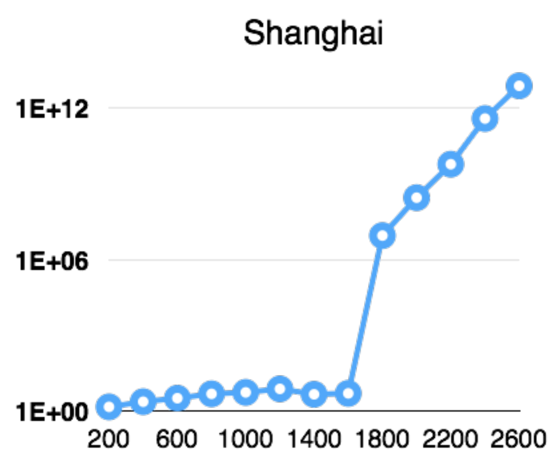

Figure 4: Shanghai only, log scale

9. Conclusion

In our project, we developed techniques which help us to determine values of η, the learning rate of the gradient descent algorithm, on the fly, allowing the algorithm to better adjust to the trends of the stock market. As we have seen, the algorithm performs well on a variety of stock data from different markets, thus resulting in relatively high wealth gains.

One possible direction of future work would be to weigh more heavily the more recent days of the look-back period than those closer to the beginning of the period, with the hope that more information can be gleaned from the more recent data. We would also hope to explore the Shanghai stock data further to try to determine the reason behind the exorbitant returns on investment.

References

[1] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[2] Lecture notes on online gradient descent. S. Arora.

[3] E. Hazan and S. Kale. On Stochastic and Worst-case Models for Investing. *Proc. NIPS*, 2009.

[4] A. Kalai and S. Vempala. Efficient Algorithms for Universal Portfolios. *J. Machine Learning Research*, 3:423-440, 2002.

[5] M. Zinkevich. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. *ICML*, 2003.