A Framework for Stock Prediction

Hung Pham, Andrew Chien, Youngwhan Lim

December 11, 2009

1 Introduction

1.1 Motivation

Stock price prediction is a classic and important problem. With a successful model for stock prediction, we can gain insight about market behavior over time, spotting trends that would otherwise not have been noticed. With the increasingly computational power of the computer, machine learning will be an efficient method to solve this problem. However, the public stock dataset is too limited for many machine learning algorithms to work with, while asking for more features may cost thousands of dollars everyday.

In this paper, we will introduce a framework in which we integrate user predictions into the current machine learning algorithm using public historical data to improve our results. The motivated idea is that, if we know all information about todays stock trading (of all specific traders), the price is predictable. Thus, if we can obtain just a partial information, we can expect to improve the current prediction lot.

With the growth of the Internet, social networks, and online social interactions, getting daily user predictions is feasible job[1]. Thus, our motivation is to design a public service incorporating historical data and users predictions to make a stronger model that will benefit everyone.

1.2 Framework description

In addition to using historical data, our framework also uses daily predictions made by users of the stock market. The framework will build two mechanisms, one which learns from historical data, and the other which learns from the user data, then combine these mechanisms into a single prediction. In particular, the framework needs to maintain the following natural requirements:

1. It must provide predictions on a daily basis and return this prediction to the users.

2. It must have comparable performance as the best users.

3. It must be stable. In particular, there is no dictator or a group of dictators.

From a high-level viewpoint, the framework will combine an objective prediction (from historical data) and subjective prediction (from user predictions). The first requirement means that the framework must employ online and adaptive algorithms and run as fast as possible to respond to a large-scale market. The second requirement means that the best users should not perform better than the algorithm, which uses data from these users. The last requirement means that the predictions are consistent over time even more information is discovered. On the other hand, it implies that there is no group of users who can decide the prediction of the whole framework (who we call "dictators"), or otherwise, the framework will become unstable as users' predictions are not stable at all. Thus, we have to control how much we want to reply on objective predictions and subjective predictions, so that the overall predictions are good, but not too subjective.

In the next section, we will describe in details our framework that can satisfy those three requirements.

[1]Certainly, it may take time to build such a system. The hard part is obtaining "serious predictions" instead of random predictions from users.

2 Prediction using historical stock information

2.1 The dataset

To make predictions, we use standard OHLCV (open, high, low, close, and volume) data that can be downloaded from financial websites[2]. We focus primarily on technology companies (Google, Yahoo, Microsoft, etc) to test our model.

2.2 Problem description

We formall formulate the stock prediction problem as follows:

> Given $n+1$ feature vectors $x_1, x_2, \ldots, x_{n+1}$ and observed labels $y_1, \ldots, y_n$ for the first n days, predict the label y_{n+1} for day $n+1$.

In our particular formulation, $x_i \in \mathbb{R}^k$ are the features on day i, and y_i is the actual price, making this a regression problem.

2.3 Objective function

Since we are basing our predictions off a feature vector, linear regression is a reasonable method to solve this problem. Using the standard linear regression method, we find a vector θ that minimizes the regret $\sum_{i=1}^{n} \left(\theta^T x_i - y_i\right)^2$, and makes the prediction $y_{n+1} = \theta^T x_{n+1}$. However, this algorithm does not capture the temporal correlation inside the data (for example, stock behavior one year ago is different from stock behavior now). Thus, we do not expect this algorithm to turn out well in practice, and empirical evidence confirms this. Nevertheless, we can still apply the idea of empirical risk minimization while trying to capture the temporal correlation.

We modify the linear regression model by allowing the vector θ to change with time. We introduce the following objective function

$$F_n = \sum_{i=1}^{n} \alpha^{-i} \ell(\theta_i, x_i, y_i) + \beta \sum_{i=2}^{n} \alpha^{-i} s(\theta_i, \theta_{i-1}).$$

Here, α is the discount factor representing how current data relates to past data, ℓ is the loss function, and $s(c,d)$ is a distance measure between c and d. In this paper, we will consider[3] $\ell(\theta, x, y) = (\theta^T x - y)^2$ and $s(c,d) = \|c-d\|^2$. The function s is introduced to gain stability by making θ_i slowly change. Next, we will apply this idea for both regression and classification models.

[2]For example, as http://www.nasdaq.com.

2.4 The classification model

Often, we care less about the actual price than we do about the change in the price from one day to the next. The algorithm above describes a regression problem, but we can perform classification by comparing tomorrow's predicted value with today's price.[4]

2.4.1 Deep Correlation Algorithm

In order to make a prediction, we minimize the objective function with some $\theta_1, \ldots, \theta_n$. This allows us to make the prediciton $y_{n+1} = \theta_n^T x_n$. To solve this optimization function, we introduce a method we call the Deep Correlation (DC) Algorithm.

Experimentally, gradient descent performs too poorly to optimize the objective function due to slow convergence. Instead, the following algorithm (Deep Correlation) uses coordinate descent to solve the optimization problem much more quickly.

```
Choose parameters $\alpha$ and $\beta$.
Choose $d$, the depth of correlation in the data.
Set $\lambda = \frac{\beta}{\alpha} + \beta$.
Repeat until convergence:
  For each $i$ from $n$ down to $n-d+1$:
    Set $z = \frac{\beta}{\alpha} x_{i+1} + \beta x_{i-1} + x_i y_i$.
    Set $\theta_i = (M^{-1})^T z$, $M = x_i x_i^T + \lambda I_k$.
  Set $\theta_{n+1} = \theta_n$.
Output $(\theta_1, \theta_2, \cdots, \theta_{n+1})$.
```

The above update rule follows by setting partial derivatives to be zero

$$\alpha x_i x_i^T \theta_i + (\beta + \alpha\beta)\theta_i = \beta\theta_{i+1} + \alpha\beta\theta_{i-1} + \alpha x_i y_i.$$

[3]This assumes a linear correlation between x and y. We can certainly make polynomial extension on the dataset to introduce more complicated correlation.

[4]We expect regression methods to work better than classification, because by changing all actual prices to 0-1 labels, we lose too much information. Since any regression algorithm can be converted into a classification algorithm, it's fine to only consider regression.

In practice, the DC algorithm is incredibly fast. It takes no more than one second before converging if we set $d = 50$ and initialize θ from previous results. This satisfies the first requirement of our framework. Moreover, after each update all $\theta_i, 1 \leq i \leq n$ fully satisfy the identity $\theta_i^T x_i = y_i$. So, the movement of θ_i does somehow represent the movement and correlation of the data.

2.4.2 Testing results

In testing mode, we introduce a threshold constant and the concept of "decidable day". A decidable day is a day whose price changes significantly compared previous day's price. For example, if we set a threshold of 1%, the decidable days are the ones where the price changes by at least 1% compared to the previous day's price. Although a good algorithm[5] should perform well on all days, the analysis of algorithms on only decidable days is also meaningful, because these are the days wehre being correct is more important.

We also preprocess the data by dividing it into discrete blocks of multiple days. The price of a block of days is the average price on those days. Thus, we aim for a long-term prediction, which is more important than a short-term one.

The following table shows the result of some basic learning algorithms. In our experiments, dividing the data into blocks does not improve these results by very much.

Stock	Decidable Days	Lin. Reg.	Log. Reg.
GOOG	51.00%	50.37%	43.87%
YHOO	52.25%	53.17%	53.39%
MSFT	30.00%	46.44%	50.42%
AAPL	61.75%	49.29%	55.87%
BIDU	71.63%	51.40%	50.44%

Figure 1. Classical methods on 800 single days, threshold 1%

The DC algorithm does not work any better than these algorithms with single-day prediction. However, when dividing data into block, we see an incredible improvement.

[5]Empirically, any algorithm consistently getting $> 50\%$ correctness can be considered good.

Stock	Decidable Days	Lin. Ext	Quad. Ext
GOOG	64.47%	**71.17%**	**72.21%**
YHOO	63.83%	**62.40%**	**65.01%**
MSFT	48.17%	**51.21%**	**51.21%**
AAPL	72.67%	**65.37%**	**65.60%**
AMZN	68.50%	**66.42%**	**67.25%**
BIDU	79.80%	**63.66%**	**66.42%**

Figure 2. DC algorithm on 600 2-days blocks, threshold 1%

Stock	Decidable Days	Lin. Ext	Quad. Ext
GOOG	80.00%	**84.58%**	**82.50%**
YHOO	75.00%	**65.33%**	**67.11%**
MSFT	62.67%	**52.13%**	**53.19%**
AAPL	81.30%	**81.97%**	**81.15%**
AMZN	81.00%	**80.66%**	**79.84%**
BIDU	86.40%	**78.24%**	**79.19%**

Figure 3. DC algorithm on 300 4-days blocks, threshold 1%

The DC algorithm performs well with many stock indexes, reaching nearly 80% when analyzing 4-days blocks with Google, Apple, Amazon, and Baidu. In contrast, it still performs poorly with Microsoft. Looking at the data, this is because Microsoft's stock price very stable. Finally, note that using quadratic expansion slightly improves linear expansion, especially with data where we have lower success rates.

2.5 The regression model

Although the classificaiton model we use is based off the regression model, the success of DC in the classification model does not carry over to the regression model, and DC performs about as well as other methods (such as linear regression) for performing actual stock price. This is because stock prices tend to change very little on a daily basis, so even the simplest algorithms (assuming the price does not change) can reach 97% to 99% correctness. In this section, we will introduce a more general version of the DC algorithm that outperforms linear regression in the long term.

2.5.1 Generalized DC algorithm

In the DC algorithm, we assume that $y_i \sim N(\mu_i, \sigma)$, where $\mu_i = \theta_i^T x_i$. Although the DC algorithm captures temporal correlation, it assumes a strong relation: the price of a day is a linear function of the feature on that day. Even when we use a quadratic expansion, the results are not improved significantly. In the general model, we modify this assumption to a more reasonable one: $y_i \sim N(\mu_i, \sigma_i)$, where $\mu_i = \sum_{j=1}^{n} x_j^T \theta_j M_{ij}$ and σ_i is some determined by some fixed function of our choice.

Under this model, the price of a day depends on all observered data (in contrast, under the standard DC algorithm the price of each day is determined by the features of that day). A reasonable choice for M_{ij} is $M_{ij} = \alpha^{i-j}$ if $i \geq j$ and $M_{ij} = \beta^{j-i}$ if $j > i$, i.e. α and β are respectively the backward and forward correlation factor. By maximizing the log-likelihood function and adding the stability requirement, we want to minimize the following objective function:

$$\sum_{i=1}^{n} \frac{1}{\sigma_i^2} \left(y_i - \sum_{j=1}^{n} x_j^T \theta_j M_{ij} \right)^2 + \sum_{i=2}^{n} \lambda_i (\theta_i - \theta_{i-1})^2.$$

As before, we find the gradient descent does not work well for this algorithm, while coordinate descent does. After some math, we get the update rules for this coordinate descent.

Let N be a matrix such that $N_{ij} = \frac{M_{ij}}{\sigma_i}$. Let $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. Let $S = N^T N$, and let D be a matrix such that $D_{ij} = (1 - \delta_{ij}) S_{ij}$, then we update

$$C_j := \lambda_j \theta_{j-1}^T + \lambda_{j+1} \theta_{j+1}^T$$

$$R_j := \left(\lambda_j I_k + \lambda_{j+1} I_k + x_j x_j^T S_{jj} \right)$$

$$\theta_j := \left(C_j + \sum_{i=1}^{n-1} \frac{y_i}{\sigma_i} N_{ij} x_j^T - \sum_{p=1}^{n} D_{pj} \theta_p^T x_p x_j^T \right) R_j^{-1}.$$

It is not hard to see that this model generalizes DC, linear regression, and weighted linear regression with appropriate choices of α, β, σ and λ. In contrast, linear regression can be considered as a greedy version of both DC algorithm and generalized DC algorithm when the θs are fixed.

2.5.2 Testing results

We test our algorithm with Google, using different α and β and compare to the standard linear regression method. We choose $\sigma_i = (n-i)^2$ to allow more variances on past data. The evaluation is based on the average linear absolute error made by each algorithm. To predict the price in the next d days, we simply shift the features back d days.

Method	Lin. Reg.	Gen. DC 1	Gen. DC 2
Tomorrow	7.037	7.412	11.021
3 days	13.778	13.505	16.003
7 days	24.762	22.076	25.840
14 days	43.654	33.862	37.474
1 months	98.269	67.779	54.406

Figure 4. Error of long-term predictions on 150 days with Google[6]

The above table suggests that Generalized DC is not better than linear regression (actually worse) when predicting tomorrow price. However, Generalized DC performs much better in long-term predictions, even when linear regression does poorly. Experiments suggest that quadratic expansion does not help Generalized DC perform better at all.

3 Prediction using user feedbacks

This section will present a method of predicting based on user's prediction so that it can satisfy requirement 2: the prediction should be relatively good compared to the best users. Although regression is more desirable, we have to restrict our framework to classification only due to time constraint. The algorithm to be presented is proved to be good (in the worst case) without any assumption about the dataset, thus we can expect that it may performed better eventually when the real dataset is fit in.

3.1 Problem Description

Assume that we have a group of n people. On day i, each person j makes a prediction $p_{ij} \in \{1, 0\}$ indicating whether the stock is up or down. We have to make prediction each day before the actual price is released at the end of the day. How to make predictions each day?

This is a typical example for boosting algorithm. We will apply the Weight Majority algorithm. Adaboost is under our consideration, but we won't really employ it until a real user dataset is obtained.

[6]The Generalized DC 1 & 2 correspond to different choice of α, β: $(0.95, 0.01)$ and $(0.8, 0.01)$

3.2 The Weighted Majority algorithm

Weighted Majority (WM) works by assigning weights for each user w_j. On day i, the prediction is made comparing P_0, P_1, in which

$$P_0 = \sum_{j:p_{ij}=0} w_j ; P_1 = \sum_{j:p_{ij}=1} w_j .$$

At the end of the day, if person j predicts wrongly, we will update his weight: $w_j = \frac{w_j}{c}$; otherwise, the weight is kept the same. It is shown that, if m_i is the total number of mistakes made by user i, and $m = \min_i m_i$, then the total number of mistakes made by the WM algorithm is no more than

$$\frac{\log n + m \log \frac{1}{c}}{\log \frac{2}{1+c}} .$$

Most importantly, this upper bound is true on daily basis with any dataset. Therefore, we can make sure that the requirement 2 is always satisfied: the prediction is as good as the best user's prediction *up to a constant factor*.

3.3 Test with sampled datasets

Although it is not a convincing evidence about the performance of WM algorithm in practice, we want to at least see how it works under some reasonable sampled dataset. Because we do not have any actual datasets of user data, we generated our own data sets from the actual prices. We created a data set such that predictions are never more than 62.4% correctness and the average correctness is $\sim$ 50%, then the result made by WM algorithm is roughly 57.5% – 65.6%.

4 Combine predictions

The previous sections present two methods of predictions: objective prediction by learning historical stock features, and subjective prediction by learning user predictions. Both methods are online and adaptive, thus satisfy requirement 1. However, requirement 3 is not always satisfied with subjective prediction, as the WM algorithm will give the best users very high weight and making them into dictators. This section will introduce how those methods can be combined together to satisfy the third requirement. Again, we restrict our model to the classification problem only.

The natural idea is to find the probabilities p_1, p_2 for the classifications made by the two algorithms, and base our final prediction on $cp_1 + (1-c)p_2$ for some constant c of our choice (or we can learn it). The probability obtained by the WM algorithm can be assumed to be $\frac{P_0}{P_0+P_1}$. The hard part is to find a probability by Generalized DC. Notice that in the Generalized DC algorithm, we assume $y_i \sim N(\mu_i, \sigma_i)$. Thus, when y_{n-1} is observed, the probability that $y_n < y_{n-1}$ is given by

$$f(y_{n-1}) = \int_{-\infty}^{y_{n-1}} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(\frac{-1}{2\sigma_i^2}(x-\mu_i)^2\right) .$$

Unfortunately, this integral has no closed form. Instead, it is represented as as

$$\frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu_i}{\sigma_i\sqrt{2}}\right)\right) .$$

Here erf, the error function, is an odd function and approximated by

$$\operatorname{erf}(x) \sim \pm\sqrt{1 - \exp\left(\frac{-(ax^2+b)x^2}{1+x^2}\right)}$$

where

$$a = \frac{8(\pi-3)}{3(4-\pi)}, b = \frac{4}{\pi} .$$

This last formulation explicitly show how the probability the price goes down is computed. Consequently, the combined probability can be computed easily. Moreover, if we choose c greater than 0.5 so that objective prediction contributes mainly to the final prediction, we can assure that there is no dictator in the framework.

5 Conclusion

This paper proposes a relatively good framework for stock prediction which can satisfy three natural requirements (with any dataset). However, this may not be the best framework in practice. Such a better one should be built when a real dataset is obtained. Nevertheless, the most important idea of the paper is to connect classical (objective) machine learning algorithms to (subjective) user feedbacks to learn complicated social structures. In the time being, no machine can really get the intelligence level of the human; therefore it is better to ask the human to learn about them and their game (unless there are machines who are more intelligent than them and can simulate them).

In the context of stock prediction, we believe that the lack of data is the major issue, while machine learners do not spend too much effort on this subject. Hedge funds and other trading cooporations should have much better datasets, but they never want to disclose it in fear that it will affect their business. In contrast, user predictions may be the best dataset for stock prediction, and if it is publicly distributed, it may encourage machine learners to attack this problem more in depth.

6 Future works

Following are a list of works that we intend to do.

1. Obtain the real user predictions. Hung has designed an online system for that purpose (see http://www.stocknet.us).
2. Make a better mechanism to combine two methods together. For example, if the prediction based on historical data suggests the price goes up, but user mechanism suggests that it goes down, should we make update on θ or the weights of users before announcing the final prediction? In the proposed framework, two methods do not really interact with each other before final predictions.
3. Develop a regression model for user mechanism.
4. Develope a better regression model for historical data prediction. Generalized DC is not totally satisfiable.

7 Acknowledgment

We would like to thank Prof. Nguyen Dinh Tho[7] and Ms. Ha Viet Phuong for their dedicated job of collecting and analysing historical stock data, detailed explanation on the structure of the stock market and suggestion on designing the platform since the last summer. We also thank Prof. Ng and Quoc Le for their help on machine learning methods and the motivation of the temporal correlation.

[7]He is currently the Director of Financial Department, Foreign Trade University, Vietnam

References

[1] Roger Lowenstein, *Efficient Market Hypothesis*, Wikipedia.org

[2] Nick Littlestone, Manfred K. Warmuth, *The weighted majority algorithm*, Information and Computation, Volume 108, issue 2, 1994.