Stock Index Prediction Using the Analysis of Twitter Sentiment

Jinyu Xia

May 11, 2016

Abstract

This paper explores the correlation between Twitter sentiment and the stock index. We first generate Twitter sentiment valences, and examine the potential weekly pattern in the sentiment valence time series by using SLT. Based on the Twitter sentiment analysis, we then apply multiple statistical and machine learning techniques to predict the stock market crashes and the S&P 500's daily trends.

1 Introduction

In Economics, Efficient Market Hypothesis suggests that the stock market is efficient enough to instantly react to all information. However, like most traditional economics theories, this hypothesis is based on one important assumption – that all individuals are rational, which in fact, is unlikely to be satisfied in most of the real-world situations. Behavioral economics believe that people's decision-making process is affected by their emotions. In other words, in the real-world economy system, individuals are not always rational. Therefore, studying and quantifying people's emotions would add to methods currently available to predict economic trend such as the stock index.

Former researchers have measured the collective public mood states derived from large-scale Twitter feeds[1]. It is found that Twitter mood states significantly improve the accuracy of Dow Jones Industrial Average (DJIA) prediction. The results show that emotions can greatly affect individual decision-making.

This paper will derive Twitter sentiment by based on the word emotion lexicon developed by Mohammad et al[4]. Based on Word Emotion Association Lexicon, developed by National Research Council (NRC) of Canada. The Twitter sentiment analysis in this paper contains three parts, which are sentiment valence generation, the weekly pattern recognition, and anomaly detection. The sentiment information will then be used not only to predict the stock index trend, but also to identify several corresponding stock crashes in 2015.

In addition, instead of using DJIA, we use the Standard&Poor's 500 (S&P 500) as the index of the stock market. The main difference between the S&P 500 and DJIA is that the S&P 500 is a market value weighted index of 500 North American Stocks that have a range of sector representations, whereas DJIA calculate only 30 North American stocks by a simple method of mathematics average. In consideration of the number of stocks covered, we select S&P 500.

2 Unsupervised Learning of Postings on Twitter

2.1 Twitter Data Raw Process

The corpus of tweets used in this project is downloaded via the Twitter API in R, which includes 8,640,000 tweets per day for almost a year. Much of the raw data is unreadable, so the first step is to clean all non-letter symbols, and filter out non-English tweets by using pattern matching and replacements in R. The second step is to filter tweets by stock-related keywords.

Tweets contents cover a wild range of topics. Note that it is meaningless to use sentiments of tweets talking about sports or TV shows to predict the stock index, and there are lots of tweets that do not even have real meaning, despite using English words. Therefore, to make the prediction of the stock index is more precise, we filter everyday tweets by keywords "stock", and several other words closely related to stock market (DJIA, Wall Streets etc.). We expect that the sentiment of tweets posted by people who care about stock market could more accurately reflect the emotions of individuals of our economic system.

After the raw filtering, about one-third of the original downloaded data remained. For each day, there are no more than 1000 tweets left. The Twitter feeds are dated from March 2015, to September 2015. About 10 days of Twitter data are missing due to the failure of a program used to download Twitter data. There are 205 days of tweets in total. Since stock market is closed on weekends and a certain holidays, there are 143 corresponding days of the S&P 500 value.

2.2 Generating Twitter Sentiment

To get the sentiment of each tweet, we have looked into two R packages which can generate Twitter sentiment. One is called `sentiment`, the other one is called `syuzhet`. At first, we prefered the package `sentiment` more since it claims to apply Bayes' Algorithm to calculate Twitter sentiment. However, we find there are several mistakes in the package source code. Hence, the reliability of the results obtained from this package cannot be guaranteed. With a more careful analysis, we believe Bayes' Theorem is a limited approach to be applied to derive the Twitter sentiment, which makes the mistakes in `sentiment` unavoidable. By comparison, using a simple voter algorithm based on a reliable dictionary might be a more solid approach, although it is a very simple algorithm.

2.2.1 Bayes' Theorem

The R pacakge `sentiment` has a lexicon of 6518 common English words. It assigns two parameters to each of the words. One tells whether this word is positive, negative, or neutral, and the other one indicates whether this word represents a strong or weak topic. Based on this lexicon, a function called `classify_polarity` calculates the polarity of a posting, by employing Bayes' theorem.

Given an input tweet, let us call it T. Suppose that we want to calculate the probability that T is positive, the formula of the Bayes' theorem is shown in Eq.1.

$$\mathbb{P}(\text{T is positive}|\text{T}) = \frac{\mathbb{P}(\text{T is positive, T})}{\mathbb{P}(\text{T})} \tag{1}$$

where $\mathbb{P}(\text{T})$ is approximated as the proportion of this words in the tweet T that are in the lexicon. The calculation of the numerator on the right hand side of Eq.1 is more complicated. It involves calculating the conditional probability:

$$\mathbb{P}(\text{T is positive, T}) = \mathbb{P}(\text{T}|\text{T is positive}) \times \mathbb{P}(\text{T is positive}) \tag{2}$$

The package function assumes that the appearance of each words in the tweet is independent. And thus the equation can be further transformed in to Eq.3.

$$\prod_{\text{word}\in\text{T}} \mathbb{P}(\text{ith word}|\text{T is positive}) \times \mathbb{P}(\text{T is positive}) \tag{3}$$

Nonetheless, this package function makes several invalid assumptions in these steps of calculation. First, the assumption of words' independence of each other is not valid in many situations, since in most text, the appearance of words are correlated with each other. Second, in the last step the package approximates the probability of tweet T being a positive tweet by using the proportion of positive words in the lexicon, overlooking the number of words in a tweet and the commonness of each word. Similarly, note that for Eq.1, the package approximates $\mathbb{P}(\text{T})$ by using the percentage of words in the lexicon of a tweet, which is also not an accurate calculation. In addition, it does not introduce the source of the lexicon used in the package, and thus we feel that the results obtained by `sentiment` are not reliable enough.

The problems of `sentiment` package reflect some difficulties of using Bayes' algorithm to derive Twitter sentiment. First of all, it is hard to calculate the probability of the appearance of a given tweet, and the probability of it being positive or negative. Additionally, to simplify the calculation we have to assume that the independence of each word, while it is a common sense that many words appears in a context dependently.

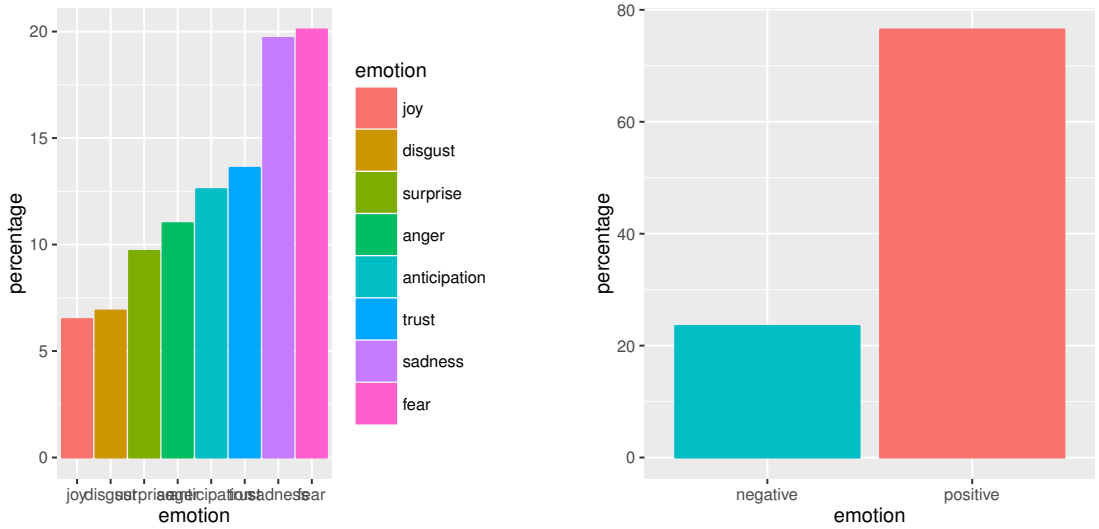
Despite that `sentiment` is a problematic package, the Bayes' algorithm is a practical approach to generate Twitter sentiment. More details of using Bayes' theorem to calculate Twitter sentiment is worthy of exploration in the future.

2.2.2 Emotion Lexicon

Our second approach for generating Twitter sentiment is to use voter algorithm based on the NRC Emotion Lexicon developed by Mohammad et al[4]. The NRC Emotion Lexicon a high-quality emotion lexicon based on manual evluation conducted by using Mechanical Turk[2]. The researchers create Human Intelligence Tasks (HITs) for terms in the lexicon, and each HIT has a set of emotion survey questions. The lexicon now include 14183 English words, and each word corresponde with eight basic emotions (anger, fear, anticipation, trust, surprise, sadness, joy, and disgust) and two sentiments (negative and positive).

The emotion scores and sentiment valence calculation of each tweet is done by using R package `syuzhet`. It uses an algorithm similar to the BoyerMoore majority vote algorithm. Since in the Twitter data clean process, non-letter symbols, including punctuations are excluded, and thus we need to treat each tweet as one sentence. Given one tweet, the presence of eight different emotions as well as a positive and negative valence in it are counted. The emotions scores and sentiment valence of a single day are the mean emotions scores and sentiment valence of hundreds of tweets on that day, respectively.

To give a simple overview of the emotion statistical data, we present an example in which the Twitter feeds on Monday, Aug 24, 2015 are analyzed. This day is an important day in 2015 stock



(a) Data Fitting

(b) Sparsity Tradeoff

Figure 1: Emotion and sentiment pie charts of tweets on August 24, 2015. This day is reported as the Wall Street's worst day in four years.

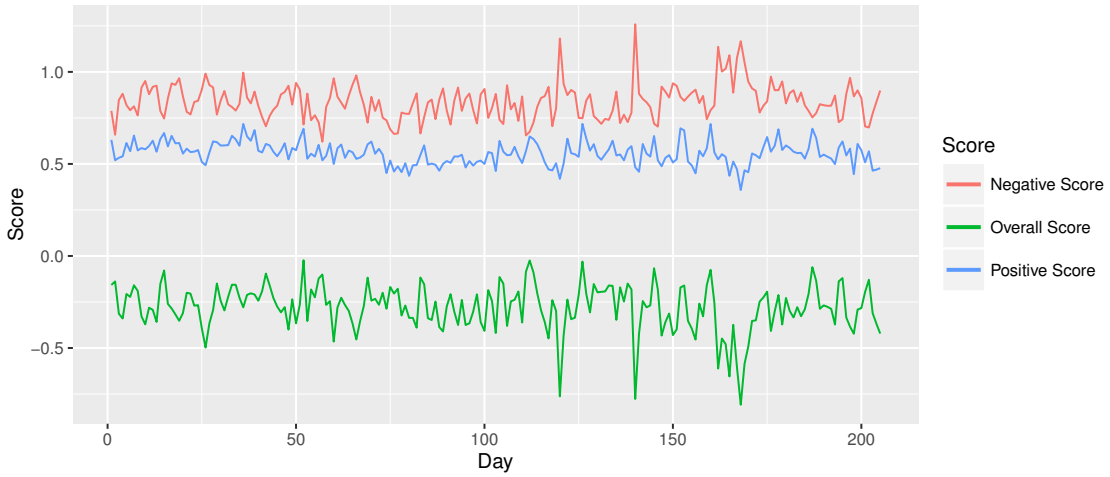


Figure 2: Twitter sentiment valence time series

market selloff. On this day, the Wall Street suffered a major starting loss as the Dow opened 1,000 points down, and it was also the S&P 500's worst day since 2011. The pie charts of the Twitter emotion show that there is more than 75% Twitter feeds present negative sentiment, and about 60% present emotions such fear, anger, sadness, and disgust. The statistical data suggest people

were expressing their negative sentiment in their Twitter posts on August 15, which is an evidence showing the existence of a relation between stock market states and Twitter sentiments.

The Twitter sentiments of Twitter feeds of 205 days are shown in Fig.2. At first glance, we can see that this time series not a homogeneous. There is fierce fluctuation in the last half of sentiment valence data. We will explore more about it in the section of anomaly detection.

3 Discovering Anomalies in Twitter Sentiment Valences

Before conducting the S&P 500 index prediction, one thing we notice is that the stock market crashes occationally. At the time when we were collecting Twitter data, a stock market crash happened to the Chinese stock market. This crash started in June 2015 and continued into July and August. It only caused a significant loss of paper wealth in China, but also set off a global rout. According to the British national daily newspaper, the Guardian, on August 24, 2015, China sent the world market into a spin, and the US stocks suffered a biggest one-day fall since 2011. The Dow Jones has ended the day down 588 points, or 3.5%; the S&P 500 lost around 3.9%, and the tech-heavy Nasdaq shed 3.8%. This 2015 market selloff inspired us to examine if the tweets we collected could convey any useful information that could be used as a warning sign of the future stock market crashes. The basic idea we have is to detect sudden spikes or dips in the Twitter sentiment time series, and probe the association between the detected Twitter sentiment "crash" and stock market crash.

3.1 What is Anomalies

The definition of anomaly, or outlier, is an element which does not follow the behavior of the majority. A concept similar to anomaly is change point, whereas the change point is not the same as an anomaly. Change point detection is the problem of discovering time points at which properties of time-series data change. Since the dataset of this project is not large enough, and also the interest of this section is to discover a certain days in which people's sentiments have violent changes, but not to find which parts of the given time series have different mean or variance.

Note that we have two time series data: the S&P 500, and Twitter sentiment calculated by NRC method. However, the interest of this section is only to detect the anomalies in the Twitter sentiment data, but not the S&P 500. There are two reasons for doing it. First, the general public has objective opinions towards what is an anomaly in the stock market, since news media is very sensitive to any abnormal violent changes in the stock market. Additionally, overall the S&P 500 is relatively a stable time series compared with most dataset. Hence, it is hard to use algorithms to detect anomalies in it.

Nonetheless, it is necessary to use academic approaches to detect anomalies of Twitter sentiment. As opposed to stock indexes, Twitter sentiment is a new concept for people, it is difficult for them to conclude on which days sentiments data have substantial changes. Thus, using professional methods to detect anomalies of the Twitter sentiment scores is essential.

3.2 Seasonal Hybrid Extreme Studentized Deviate test

In this section, we use an R package called `AnomalyDetection`, which could automatically detect anomalies in big data in a practical and robust way. The algorithm used to detect anomalies in Twitter sentiment valence is called Seasonal Hybrid Extreme Studentized Deviate (ESD) test. ESD

test is also known as Grubb's test, which is a statistical test used to detect outliers in a univariate data set assumed to come from a normally distribued population. The null hypothesis of EST test is that there is no outliers in the data set. EST test detects one outlier at a time, until there are no more outlier detected. The EST test's statistic is defined as:

$$G = \frac{\max_{i=1,\ldots,N} |Y_i - \bar{Y}|}{s} \tag{4}$$

where $\bar{Y}$ and s denoting the sample mean and standard deviation, respectively. The hypothesis of no outliers is rejected at significance level α if:

$$G > \frac{N-1}{\sqrt{N}} \sqrt{\frac{t^2_{\alpha/(2N),N-2}}{N-2+t^2_{\alpha/(2N),N-2}}} \tag{5}$$

Notice that ESD test is based on the assumption of normality, but many real data are not normally distributed. Real data, however, may have some seasonality. Therefore, one approach to deal with non-normal data is to first fit its seasonal adjustment. This is where Seasonal-Trend Decomposition (STL) by Loess comes in[3]. STL decomposes seasonal time series into trend, seasonal, and remainder, by using locally-weighted regression, or LOESS. STL has a simple design that consists of a sequence of applications of the loess smoother. Other features of STL are specification of amounts of seasonal and tread smoothing that range.

The `AnomalyDetection` package uses the Seasonal Hybrid ESD algorithm, which combines seasonal decomposition with robust statistical methods to idnetify local and global anomalies. This is might be an ideal package applied to our Twitter sentiment time series, since at Twitter, a distinct seasonal patterns could be observed in most of the time series. An anomaly can be detected when this point is extreme enough to extend beyond the natural seasonal variation in the time seires (Twitter calls these "global anomalies"). Also this algoirthm could be used to idenfiy "local anomalies", which are small variations on the seasonal trend, but which do not extend the beyong the usual range of values.

3.3 Detection: Warning Sign of Stock Market Crashes

We applied the `AnomalyDetection` package to both Twitter positive and negative valence time series we obtained before. Interestingly, for the positive valence time series, there is no anomaly found, but there are four anomalies detected in the negative valence time series. The date of the anomalies are listed as below:

Table 1: Twitter Negative Sentiment Valence, Anomalies

	Date	timestamp	anomalies	expected value
1	2015-06-29	01:00:00	1.181598	0
2	2015-07-27	01:00:00	1.259012	0
3	2015-08-23	01:00:00	1.076377	0
4	2015-08-24	01:00:00	1.166600	0

We mark these four anomalies in the negative valence time series. As we can see, these four points do have obvious deviation from the behavior of the majority. The corresponding points of the

S&P 500 is also market, and as we can see, on all of the days of the sentiment anomalies, there are a sudden drop of the S&P 500 index. Also notes that the distribution of Twitter negative valence time series starting from July 2015 becomes very strange and unusual. In regard of observation, a further change-point analysis is probably worth conducting on the twitter sentiment time series, which may be helpful for predicting stock market crashes.

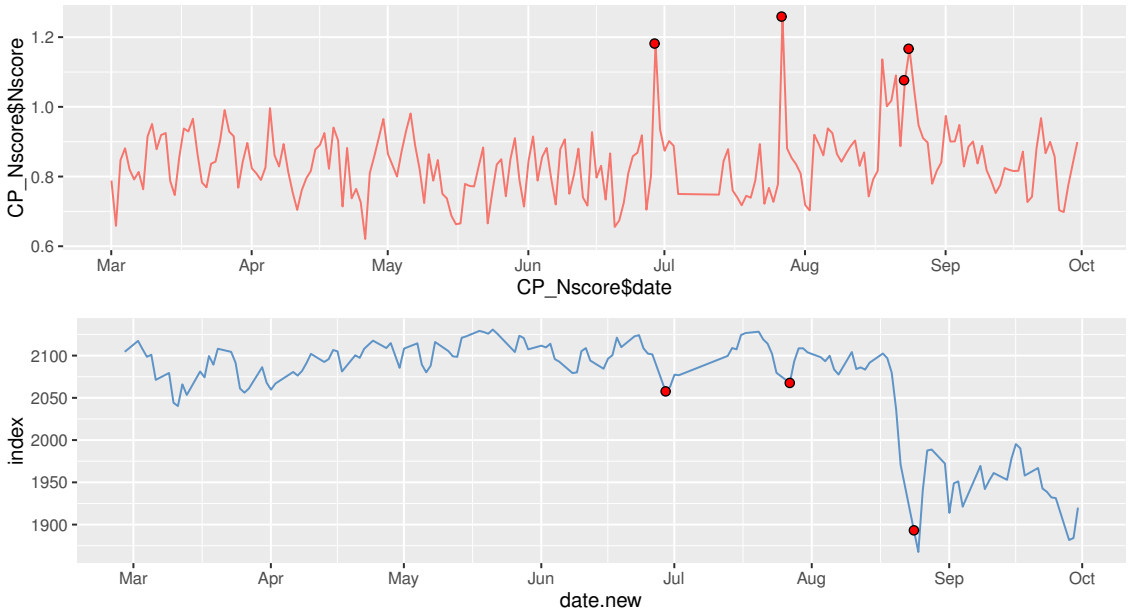


Figure 3: Comparison

4 Stock Market Index Trend Prediction

In this section, we will explore several machine learning supervised models to examine the potential relationships between the derived Twitter sentiment and the S&P 500 index. We treat the problem as a classification task, whereas the label is the daily up and down of the S&P 500 index. In section 4.1, we conduct Least features that may help to predict the up the daily trend of the stock index. In section 4.2, we use regularized methods, Lasso and Rigide, to improve the generalization of the learned model. We've also tried logistic regression and support vector machines (SVM) on the S&P 500 index prediction. Logictisc regressio yields error rates slight worse than LS, and SVM yield a very undesirable results, which is probably due to the small size of features we have.

4.1 Least Square Binary Classification

We first use the Least Square (LS) binary classification. The label y is the daily trend of the S&P 500 index. A daily up and down is labeled as 1 and -1, respectively. The original features contain the positive and negative Twitter sentiment valence from the current day to 5 days ago;

also included in the feature sets are the labels from one day ago to 5 days ago. Thus, the number of features n is 17.

A common method for estimating the performance of a classifier is cross-validation (CV). We divide the dataset into 9 equal sized subsets, each of 14 data (so that we only use 126 out of 138 days' of Twitter sentiment). Note that if the dataset is not time series, we can simply use 8 sets of the data to choose the weights, then use the weights to predict the labels of the remaining "hold-out" set, and compute the number of mistakes made on this hold-out set and divide that number by 14 (the size of the set). The last step is to repeat this process 9 times (for the 8 different choices of the hold-out set) and average the error rates to obtain a final estimate.

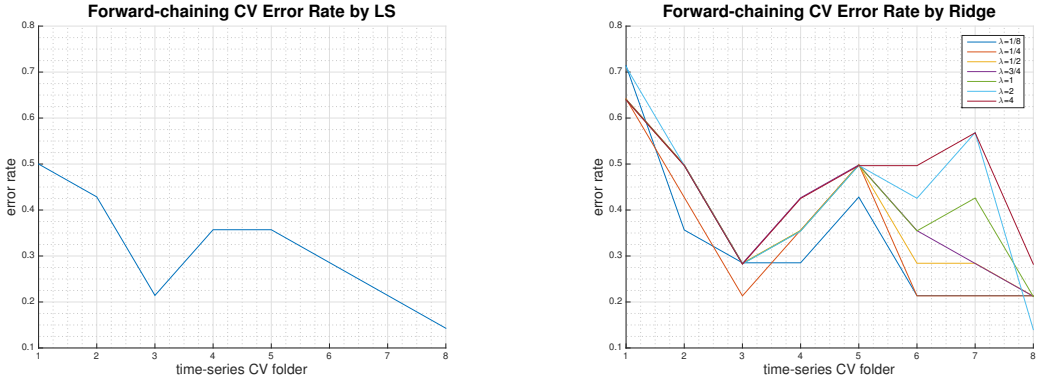
However, since our dataset is a time series, we cannot use the future data as the training set while the past data make up the test set. Hence, the estimation of the model performance is slightly different here. We still use 9 folders, but we cannot cross validate the error rate by repeating the process of randomly splitting the set of data into 9 folders of size 14. Alternatively, we can only use the folders that contain the past data to predict a hold-out folder that contains the future data. An approach that is sometimes more principled for time series is forward chaining, where the procedure is shown below:

- folder 1: training [1], test [2]
- folder 2: training [1 2], test [3]
- folder 3: training [1 2 3], test [4]
- ...
- folder 8: training [1 2 3 ... 8], test [9]

This forward chaining method enables us to model on past data and predict on forward-looking data. The error rates of the 8 test folders are: 0.5000, 0.4286, 0.2143, 0.3571, 0.3571, 0.2871, 0.2143 and 0.1429. As we can tell, the overall trend is that as the size of the training set grows, the error rate becomes smaller. When the test set is the last fold, and all the "past" folders are used as the training set, the error rate is the smallest. This makes sense since more training set give us more information for the model fitting. In the best situation, there are 2 misclassifications of the daily trend of the S&P 500 index out of 14 days.

4.2 Refinement: Regularized Least Square Optimization

One challenging part of our prediction problem is how to choose good features. In section 4.1, the size of the initial feature sets n is 17, whereas 12 of them are Twitter positive and negative valence, and 5 of them are past daily trends of the stock index. Among the 12 valence features, there may exist potential data dependency. The correlation matrix of the 12 features shows that the positive valence score of current day and one day ago have the largest correlation, which is 0.47. A few other couple of features also have correlation higher than 0.4. Although they are not desperately high correlations, but fitting 12 similar features together need to certainly be handled carefully, so that we do not overfit the training data. In this section, we will discuss two common regularized least squares (RLS) to improve the generalization of our model by constraining it at training time.



(a) LSData Fitting

(b) Performance of Ridge Regression

Figure 4: Least Squares vs. Ridge

4.2.1 l_2 regularizer: Ridge

We will first try Ridge (aka Tikhonov) regularization. Ridge is a RLS problem with ℓ_2 regularizer:

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_2 \tag{6}$$

where λ is for penalizing the model if the sum of the 2-norms of the fitted weights get too high. In other words, Ridge regression can avoid over-fitting by regularizing the weights to keep them small. Eq.6 has a closed-form algebraic solution $\hat{\boldsymbol{\beta}} = (\boldsymbol{X^T X} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$. Since n is not too large, we can compute the solution directly. The performance of the ridge parameters is shown in Fig.4(b).

Fig.4 shows 7 lines of error rate, corresponding with different choices of λ ranging from 1/8 to 4. The overall trend of these lines is similar to the shape of the error rate line in Fig.4(a). The regularization achieves its best performance when we all 8 folders are used as the training set, and $\lambda = 2$ yields the best error rate 0.1429, which the same as the LS method.

As we can see, Ridge regression does not improve the performance of our model. This is due to two reasons. First, the error rate of the LS method has already achieved a small error rate that is hard to exceed. Second, the size of our features is not desperately large; thus Ridge does not serve a useful purpose here. However, the weights fitted by Ridge when $\lambda = 2$ is a good candidate set of parameters for future prediction, since it can avoid potential overheating.

4.2.2 l_1 Regularizer: Lasso

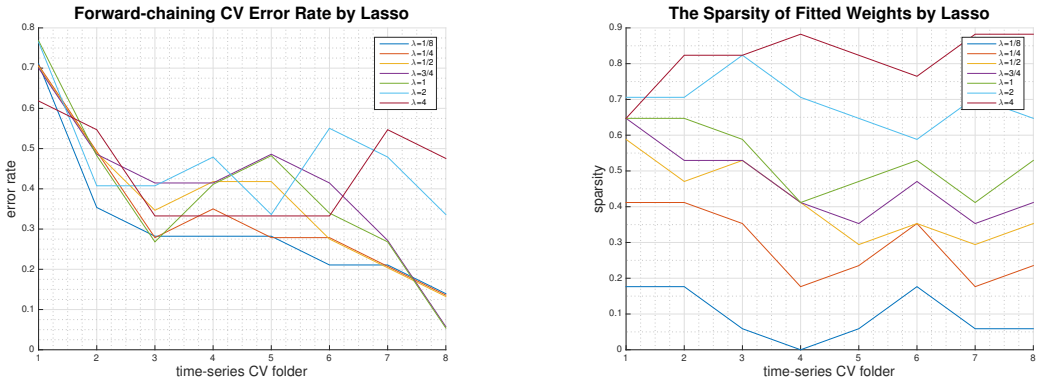
The regularized least squares optimization problem with ℓ_1 regularizer is called the Lasso in machine learning:

$$\min_{\boldsymbol{\beta}} \|\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_1 \tag{7}$$

The Lasso has some very interesting properties. Most notably, the solution to the optimization above usually has many of the elements of $\boldsymbol{\beta}$ set to zero. It tends to put weight only on elements of $\boldsymbol{\beta}$ that correspond to the columns of $\boldsymbol{X}$ that are most predictive of $\boldsymbol{y}$. The larger λ is, the sparser

the solution $\hat{\boldsymbol{\beta}}$. Unlike Ridge, we do not have a closed-form solution for the Lasso. The solution to Eq.7 is given by the "soft-threshold" operation:

$$\beta_j = sign(y_i)(\|y_i\| - \lambda/2)_+, \tag{8}$$



(a) Data Fitting

(b) Sparsity Tradeoff

Figure 5: Performance of Lasso Regression

Despite of the nonexistence of the closed-form solution to the Lasso, iterative solvers can be used to obtain good approximate to the solution at a lower cost. We choose to use the Landweber iteration to the optimization. It could be proved that the bound optimization algorithm to approximate the lasso solution is given by

$$\beta_{k+1} = sign(\boldsymbol{z}_k)(\|\boldsymbol{z}_k\| - \tau\lambda/2)_+, \tag{9}$$

where $\boldsymbol{z_k} = \boldsymbol{\beta_k} - \tau \boldsymbol{X^T}(\boldsymbol{X}\boldsymbol{\beta_k} - \boldsymbol{y})$. The performance and the weights sparsity of the Lasso is shown in Fig.5. The figure indicates that $\lambda < 1$ yield a lower error rate. When λ is 3/4 and 1, the error rate is 0.0714 (i.e. There is only 1 misclassification in the test set), which is a result not achieved by LS method. In addition, Fig.5(b) shows that the Lasso with $\lambda = 1$ or 3/4 select only half of the original features. Based on this information, we can conclude that the Lasso help improving the generalization of our model and avoid overfitting.

5 Conclusion

In this paper, we have described our exploration of the association between sentiment in Twitter posts and economic indicators, S&P 500 market index. We discusseed several potential methods of retrieving Twitter sentiment, and use the one based on NRC Emotion Lexicon developed by Saif Mohammad et al. We have seen that the anomalies of the negative twitter valence is a good indication of great stock market crash. We have also explored the prediction performance of S&P 500 daily trend by using machine learning techniques such as LS, Lasso, Rigid, logistic regression, and SVM, and presented the results of the models which yield decent results. The best prediction result is seen in Lasso regression (λ = 3/4 or 1) by using all the first 8 folders as training set to

predict the last folder of the data. In this setting, there is only one missclassification of the total 14 response variables. Based on our data analysis and conclusion, several improvement could be made to increase the prediction rate. One possibility is to increase the data set; due to the time constraint, the data set size of our project is no more than 200. Another promising adaptation to made is to find more features that might be helpful for stock market index prediction. Those could be features not related to twitter sentiment, such as unemployment rate and oil prices.

References

[1] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

[2] Chris Callison-Burch. Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 286–295. Association for Computational Linguistics, 2009.

[3] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, 1990.

[4] Saif M Mohammad and Peter D Turney. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 26–34. Association for Computational Linguistics, 2010.