An Efficient System for Detecting Outliers from Financial Time Series

Carson Kai-Sang Leung, Ruppa K. Thulasiram, and Dmitri A. Bondarenko

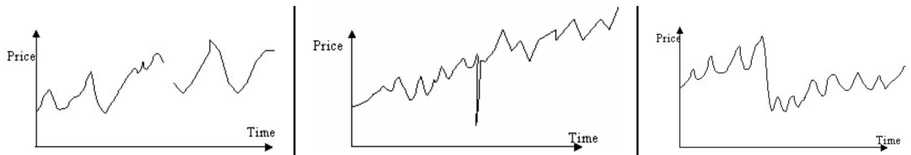
The University of Manitoba, Winnipeg, MB, Canada
{kleung, tulsi, umbonda1}@cs.umanitoba.ca

Abstract. In this paper, we develop an efficient system to detect outliers from real-life financial time series comprising of security prices. Our system consists of a data mining algorithm and a statistical algorithm. When applying each of these two algorithms individually, we observed its strengths and weaknesses. To overcome the weaknesses of the two algorithms, we combine the algorithms together. By so doing, we efficiently detect outliers from the financial time series. Moreover, the resulting (processed) datasets can then be used as input for some financial models used in forecasting future security prices or in predicting future market behaviour. This shows an alternative role of our outlier detection system—serving as a pre-processing step for other financial models.

Keywords: Databases, data mining, computational finance.

1 Introduction

Data mining refers to the search for implicit, previously unknown, and potentially useful information or patterns that might be embedded in data. Many of the existing studies focused on finding patterns that apply to the majority of items in the dataset [1, 2, 8, 10, 11, 14]. However, patterns that apply to the minority of items can also be interesting and important. For example, a rare event could be an indication of some unusual, suspicious, or criminal activities. Hence, several studies focused on *outlier detection* [6, 7, 9, 13], which aimed to analyse and find these exceptional activities from datasets like the performance statistics of professional athletes, workers' compensation data, and medical test data. Moreover, outlier detection could be used in various application areas such as e-commerce and finance. In this paper, we show how outlier detection can be applicable to financial data in an emerging cross-disciplinary area of research, known as *computational finance*, that addresses problems in finance or business (e.g., option pricing, portfolio management) by using advanced scientific computing or data mining techniques. In this area, several models have been developed to forecast future security prices and to predict future market behaviour. These models usually rely on standardised historical data, and are sensitive to data variations. *Any unusual noise (i.e., outliers / data polluters) present in the data may lead to incorrect forecast or prediction.* To ensure good prediction of price behaviour, many models require historical price data over a long



(a) Missing data **(b)** Short-lived sudden price changes **(c)** Long-lived sudden price changes

Fig. 1. Data polluters

period of time (e.g., $\geq$ 10 years) for thousands of securities. Similarly, to ensure accuracy in applications like option pricing and risk management, several models require the input price series to be free from noise. Although the price series are normally obtained from reliable sources (e.g., Bloomberg), the series might still contain some *data polluters* (as shown in Fig. 1) such as: (a) missing data[1], (b) short-lived sudden price changes[2], and (c) long-lived sudden price changes[3].

Outliers can significantly influence financial model outputs (e.g., the forecast or prediction). Hence, to achieve better forecast or prediction, we need a system for detecting and eliminating outliers as well as pre-processing data. Algorithms in such a system should: (i) run efficiently on large datasets; (ii) detect both missing data and short-lived sudden price changes as *outliers*, and eliminate them; (iii) accommodate and ignore long-lived sudden price changes (as they should *not* be considered as outlier); and (iv) allow user input and control.

Over the past few years, some studies [12, 15, 16] in finance have suggested that noise/outlier detection from time series forms a fundamental problem. They often require data for their financial models to be free from noise. However, these studies mainly dealt with portfolio selection but *not* focused on outlier detection; they did not mention how to remove noise either. To detect outliers, other related works have been proposed. Some of them used statistical techniques like principal component analysis [5, 15] and independence component analysis [3], while some others used data mining techniques like clustering[4] [4] and anomaly detection [6, 7, 9]. However, most of these works did not use both data mining and statistical techniques. In contrast, we apply both techniques in this paper.

Our key contribution of this paper is the development of an efficient system for detecting outliers from financial time series. More specifically, our technical contributions of this paper are as follows:

[1] They could be caused by market closure in observation of a bank holiday, the stock not being traded on that day, or incomplete information at the sources.

[2] They could be caused by a price recording error or by market over-reaction.

[3] They are usually caused by a *stock split*—a situation when the stock price moves far up (under the normal condition), the issuing company splits the stock into two (or more) so as to keep up with the market demand while at the same time making it affordable for common investors by reducing the original price to half (or lesser).

[4] An item in the data is an *outlier* if it does not belong to any clusters.

- We develop a *data mining algorithm*, which uses a distance-based outlier detection technique to detect outliers from time series of security prices. The algorithm checks the values of security prices within certain distance so as to verify if the current price is an outlier.
- We also develop a *statistical algorithm*, which uses normal distribution properties of data to detect outliers. Moving averages are used in this algorithm to render the algorithm efficient.
- Due to their varying nature and properties, the above two algorithms may detect different outliers from the same time series. However, they are complementary. So, by putting them together into our outlier detection system, most (if not all) outliers can be effectively detected and removed. The resulting time series (i.e., the processed financial data) can then be used as input to existing financial models (e.g., neural network architecture) for a more accurate forecast of future security prices, a more accurate prediction of future market behaviour, and more accurate computation of option prices. This demonstrates an alternative role of outlier detection technique (as a pre-processor than as a stand-alone tool for obtaining insight into data distribution).

This paper is organised as follows. The next section describes our outlier detection system for financial applications. Section 3 shows experimental results. Finally, conclusions are presented in Section 4.

2 Our Proposed Outlier Detection System

In this section, we start describing our proposed outlier detection system, which consists of two phases. In Phase I, we identify the missing prices in a given dataset (i.e., financial time series), and substitute them with a new price. The new price is calculated in such a way that it ensures consistency with its neighbouring prices. This is done to avoid the introduction/generation of "artificial" outliers (noise). In Phase II, we run both a *data mining algorithm* and a *statistical algorithm* to detect those short-lived sudden price changes (i.e., outliers) based on data mining (or more specifically, distance-based data mining) and statistical approaches, respectively. We note that the execution of the data mining algorithm does not depend on the result of the statistical algorithm, and vice versa. Hence, by combining these two underlying algorithms, we nullify their individual weaknesses and speed up the outlier detection process.

2.1 Phase I: Identifying Missing Data

The goal of Phase I of our system is to identify gaps (i.e., missing data/prices, such as those depicted in Fig. 1(a), in the financial time series) and to fill them with new prices. With this respect, we develop a gap-identification algorithm based on the following realistic assumptions: There are five business days every week, and data for each business day are available (regardless whether it is a holiday or not). The algorithm, which has a linear complexity with a single scan

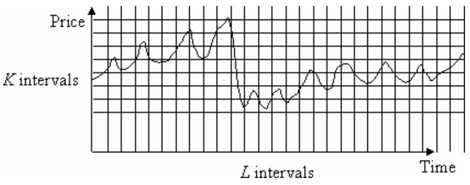


Fig. 2. The search space for the DetectOutliersDM algorithm

of the dataset, runs as follows. It scans the whole dataset once, and divides price items on a weekly basis. If any item for a certain weekday is missing, it is substituted with an item generated by a function, called *NewPrice*, which is used to obtain (i) an approximate value for the missing item or (ii) a new value of an item that is considered an outlier. The complexity of function *NewPrice* is linear with respect to the size of interval used for calculating the average. Here, we assume that items are normally distributed. Prices that are close (based on the date) are expected to influence each other to a greater extent than those prices that are far apart. So, in abstract terms, the new price can be computed based on the following equation:

$NewPrice = \left[\max\left(\sum_{i=1}^{n} I_{t-i} w_{ui}, 0\right)/2n\right] + \left[\max\left(\sum_{i=1}^{m} I_{t+i} w_{di}, 0\right)/2m\right]$,
where I_t is an item in the time series, w_{ui} are the weights of the preceding (upstream) items, and w_{di} are the weights of the following (downstream) items; m and n are numbers of neighbouring data items in the downstream and upstream directions, respectively.

2.2 Phase II: Detecting Short-Lived Sudden Price Changes

Once the missing prices are identified, we can apply Phase II of our system to detect outlying prices from the financial time series. For this phase, we develop a data mining algorithm (*DetectOutliersDM*) and a statistical algorithm (*DetectOutliersStat*), and then effectively combine them into our system.

The Data Mining Algorithm. An algorithm in Phase II is a distance-based data mining algorithm called **DetectOutliersDM**. This algorithm is based on the FindAllOutsM algorithm [7], which uses a distance-based notion of outliers to detect outlying items. According to this notion, an item I_t in a dataset is an *outlier* if most items in the dataset lie at a distance greater than a user-defined threshold D from I_t.

The key idea of our DetectOutliersDM algorithm can be described as follows. The financial time series can be represented in the two-dimensional space with prices along the y-direction and time along the x-direction. Similarly, we divide the space between the maximum and minimum item values (i.e., price values) of the dataset into K equal intervals along the y-axis, where K is a user-specified constant and the size of each interval is (MaxPrice − MinPrice)/K. The space between the first and the last dates is divided into L intervals along the x-axis. We then map these L time intervals into the K price intervals so that each cell is a square and uses the same units of measurement. This scenario is depicted

```
DetectOutliersStat
(1) Calculate the sum and the sum of squares of the first $i$ items that follow the first item in the
    set (where $i$ is a constant or a user-defined value).
(2) For each item in the dataset:
    (a) Calculate the mean and the standard deviation for the current item based on the sum and
        the sum of squares. There are two means and two standard deviations for each item (based
        on items in the upstream and downstream directions).
    (b) Calculate item rankings, and compare them to some predefined threshold (which is either
        a constant or a user-defined value). An item is an outlier if both rankings are greater
        than the threshold. If the current item is marked as an outlier, replace it with a new item
        generated by NewPrice (as described in Section 2.1).
    (c) Update the sum and the sum of squares for the next item in the set.
```

Fig. 3. A Skeleton for the DetectOutliersStat algorithm

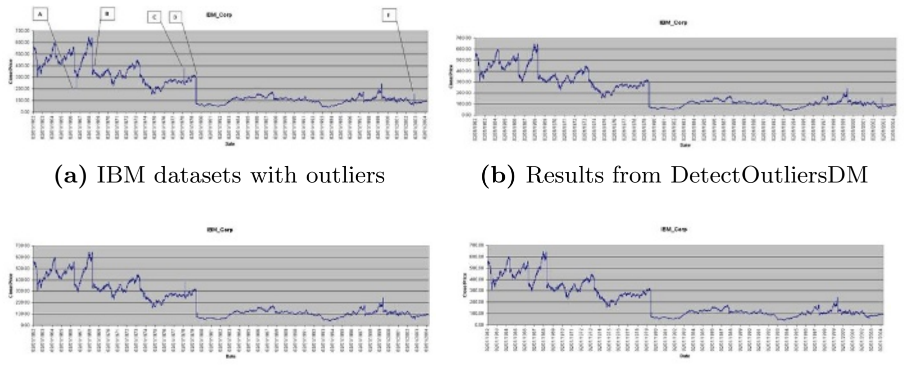
in Fig. 2. The distance that defines the neighbourhood for each item is then equal to:

$$D = (\text{MaxPrice} - \text{MinPrice})/K \times 2\sqrt{2}.$$

We use the Euclidean distance for calculation. With this setting, a price item is considered an *outlier* if it has insufficient number of neighbours within distance D (i.e., within a cell with low item density). For the financial time series (with each price quantised into a two-dimensional space), the complexity of this cell-based algorithm depends on the number of price items N in the dataset and the number of cells in the space. More precisely, the algorithm has a linear complexity with respect to N.

As a preview, we will show in Section 3 that our distance-based data mining algorithm is effective in detecting outliers (especially in detecting those short-lived sudden price changes that are lying away from the normal price range).

The Statistical Algorithm. The second algorithm in Phase II is a statistical based algorithm called **DetectOutliersStat**. Here, we make an assumption that items are normally distributed. Although it might not be true for the whole dataset, this assumption usually holds for many short continuous sub-groups in the dataset. The DetectOutliersStat algorithm is based on a statistical observation that most items are located within three standard deviations from the mean (or average). Thus, if an item is 10 standard deviations away from the mean, it is very likely to be an outlier. The distance that measured in standard deviations from the mean is defined as *item ranking*. Each item I_t has two such rankings: One ranking is based on the items that precede I_t (i.e., upstream) and another ranking is based on the items that follow I_t (i.e., downstream). In order to be considered an *outlier*, an item I_t needs to have both rankings greater than some specified thresholds. To improve efficiency, we avoid calculating the mean for each item from scratch; instead, we use the moving averages. Since the algorithm works in a sequential manner, the averages (means) for the current price item can be calculated by adjusting the averages for the upstream items. The complexity of this algorithm is linear, and it requires only a single scan of the entire dataset. The items that are marked as outliers are replaced with new items generated by the *NewPrice* function described in Section 2.1. Fig. 3 shows a skeleton of this statistical algorithm.



(a) IBM datasets with outliers **(b)** Results from DetectOutliersDM

(c) Results from DetectOutliersStat **(d)** Results from combined algorithm

Fig. 4. Experimental results on the IBM Corporation dataset

3 Experimental Results

We ran our proposed system over sets of real (historical) security price time series, which were originated from `www.yahoo.com`. To test the performance and effectiveness of our system, we added some "artificial" outliers to the time series. Regarding our system, the interface was built using Microsoft Excel with Visual Basic for Applications, and the underlying mining engine was implemented using Visual C++. As security datasets are processed one at a time, we assume that a dataset for a particular security will fit entirely into memory. This realistic assumption holds because in an extreme case, there are only about 26,000 prices for each security if day-to-day historical data are available for the past 100 years.

In the experiments, we tested our proposed system using various datasets (e.g., financial time series comprising of security prices for IBM Corporation, Boeing, Microsoft Corporation, etc.). The results were consistent. So, for lack of space, we only show the experimental results on the time series for IBM Corporation for the period 1962–2004. For this dataset (Fig. 4(a)), there are several important items that are worth special mentioning: A is a single outlier that is outside of the normal range of that interval in the time series. B is *not* an outlier; it is just a point where the stock price suddenly drops (due to some natural factors such as a change in economic situation or a stock split). C is a double-itemed outlier (where the two price items are very close to each other), which does not fall outside of the normal range of the dataset. D (which occurs not too far after the long-lived sudden price change) and E are single outliers that are within the normal range of the time series.

We first applied **DetectOutliersDM** to the IBM time series. Results in Fig. 4(b) show that our algorithm was able to successfully detect and remove outliers A and C, while leaving non-outlier B intact. An advantage of this algorithm is its effective removal of double-itemed (or multi-itemed) outliers. This stems from the fact that the algorithm relies on the number of neighbours of a given data item to determine an outlier. Having only a small number of neighbours would be a good indication of an outlier. However, the algorithm failed to identify outliers D and E, because these two outliers were within the normal range of the dataset and they had a large number of items in their surrounding neighbourhood; hence, the algorithm did not see them as outliers.

We then applied **DetectOutliersStat**, which calculates moving averages for each item. As shown in Fig. 4(c), our algorithm successfully detected and removed outliers A, D and E, while leaving non-outlier B intact. However, the algorithm failed to identify double-itemed outlier C. It is because this algorithm used statistical methods based on averages and standard deviations. When there were several outliers that are close to each other, they would influence each other's averages and standard deviation. This would lead to a lower ranking of the individual items in the time series, and hence failed to correctly identify outliers.

To summarise, the above experimental results show the strengths and weaknesses of DetectOutliersDM and DetectOutliersStat. Due to their varying nature and properties, the algorithms may not necessarily detect and remove the same outliers from the financial time series. Instead, they may detect different outliers caused by short-lived sudden price changes. For example, both algorithms were able to detect outlier A (a single outlier that is *outside* of the normal range) and leave non-outlier B intact. However, outlier C (a *multi-itemed* outlier) was only detected by DetectOutliersDM; outliers D and E (outliers that are *within* normal range) were only detected by DetectOutliersStat.

Observing the strengths and weaknesses of these two algorithms, we finally put the two together into our proposed system. Results in Fig. 4(d) indicate that our system comprising of both algorithms successfully detected outliers A, C, D and E while left non-outlier B intact. This shows the *effectiveness* of our system.

Next, let us then turn our attention to the *efficiency* issue. Experiments were conducted using the above IBM dataset on a single processor machine with 512 MB of operating memory. Results show that the execution times for both algorithms were short ($\approx$ 1 second) and approximately the same.

As the output from our system (the resulting/processed datasets) can be used as input for financial models used in forecasting future security prices or in predicting future market behaviour, we plan to conduct some experiments to study the improvement in the forecasting of future security prices and the computation of option prices.

4 Conclusions

We developed an efficient system to successfully detect outliers from financial time series. Our system consists of two phases: Phase I identifies the missing data, and replaces them with new prices that are consistent with their neighbouring prices; Phase II detects short-lived sudden price changes. We developed two algorithms for this second phase. Our data mining algorithm, called DetectOutliersDM, uses a distance-based approach to detect outliers (especially those items having insufficient neighbours and those multi-itemed outliers). Our statistical algorithm, called DetectOutliersStat, uses moving averages of each item to detect outliers (especially those outlying items that are within the normal range and those single-itemed outliers). While both of these two algorithms are effective in detecting most outliers, there exist some outliers that are detected by only one of the algorithms. We observed and understood the strengths and weaknesses of the two algorithms, and we put them together. Then, an item in the time series is considered as an outlier if it is detected by one of the algorithms. Consequently, more outliers can be effectively detected and removed. This, in turns, leads to cleaner time series (i.e., with less noise).

This paper shows a confluence of various disciplines—namely, data mining, statistics, and finance. It also shows an additional applicability of outlier detection techniques. To elaborate, many existing outlier detection algorithms are generally served as stand-alone tools for obtaining insight into data distribution. In contrast, our proposed outlier detection system can be served as a preprocessing step for other algorithms (e.g., financial models for forecasting future security prices, predicting future market behaviour, and/or pricing complex financial instruments such as derivatives).

Acknowledgement. This project is partially sponsored by Science and Engineering Research Canada (NSERC) and The University of Manitoba in the form of research grants.

References

1. Agrawal, R., Ghosh, S., Imielinski, T., Iyer, B., Swami, A.: An interval classifier for database mining applications. In: Proc. VLDB 1992. 560–573
2. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Advances in Knowledge Discovery and Data Mining (1996) ch. 12
3. Back, A.D., Weigend, A.S.: A first application of independent component analysis to extracting structure from stock returns. World Scientific - IJNS **8** (1997) 473–484
4. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. KDD 1996. 226–231
5. Jolliffe, I.: Principal Component Analysis, Springer-Verlag (1986)
6. Keogh, E., Lonardi, S., Chiu, B.Y.: Finding surprising patterns in a time series database in linear time and space. In: Proc. KDD 2002. 550–556
7. Knorr, E.M., Ng, R.T.: Algorithms for mining distance-based outliers in large datasets. In: Proc. VLDB 1998. 392–403
8. Lakshmanan, L.V.S., Leung, C.K.-S., Ng, R.T.: Dynamic mining of constrained frequent sets. ACM TODS **28** (2003) 337–389
9. Leung, C.K.-S.: Evaluation of data mining opportunities at Workers' Compensation Board. Research report, Workers' Compensation Board of BC, Canada (1998)
10. Leung, C.K.-S., Khan, Q.I., Hoque, T.: CanTree: a tree structure for efficient incremental mining of frequent patterns. In: Proc. ICDM 2005. 274–281
11. Omiecinski, E., Savasere, A.: Efficient mining of association rules in large dynamic databases. In: Proc. BNCOD 1998. 49–63

12. Park, J.: Modern portfolio theory and its application to hedge funds: Part II. MFA Reporter (Aug. 2001) 1–2, 4, 12–13
13. Schwertman, N.C., Owens, M.A., Adnan, R.: A simple more general boxplot method for identifying outliers. Elsevier - CSDA **47** (2004) 165–174
14. Srikumar, K., Bhasker, B., Tripathi, S.K.: MaxDomino: efficiently mining maximal sets. In: Proc. BNCOD 2003. 131–139
15. Utans, J.W., Holt, T., Refenes, A.N.: Principal components for modelling multi-currency portfolios. In: Proc. NNCM 1996. 359–368
16. Victoria-Feser, M.-P.: Robust portfolio selection. Working paper 2000.14, University of Geneva, Switzerland (2000)