TimeExplorer: Similarity Search Time Series by Their Signatures

Tuan Nhon Dang and Leland Wilkinson

University of Illinois at Chicago

Abstract. The analysis of different time series is an important activity in many areas of science and engineering. In this paper, we introduce a new method (feature extraction for time series) and an application (TimeExplorer) for similarity-based time series querying. The method is based on eleven characterizations of line graphs presenting time series. These characterizations include measures, such as, means, standard deviations, differences, and periodicities. A similarity metric is then computed on these measures. Finally, we use the similarity metric to search for similar time series in the database.

1 Introduction

Time series analysis is used for many applications, such as economic stock market analysis, census analysis, and inventory studies. In the last decade, we have observed the dramatic changes in the way that researchers analyze time series. This direction is also presented in many data mining and knowledge discovery papers.

Extracting features from time series data is a fundamental problem. These features can be used to query similarity-based pattern querying in time-series databases [1], cluster time-series data [2], classify time-series data [3], or detect anomalous subsequences in time series [4]. Existing similarity time series search methods roughly fall into three categories depending on how we present and compare time series: distance metric methods, dimensionality reduction methods, and interest point methods.

1) **Distance metric methods:** Many methods have been proposed for calculating the distance between time series. Some of the most used are the Euclidean distance, the Manhattan distance, and Dynamic Time Warping (DTW) [5]. These methods are not robust to outliers. Consequently, edit distance methods (the idea was borrowed from matching strings) enable matching by ignoring the dissimilar parts of the given time series [3,6].

2) **Dimensionality reduction methods:** These methods compress time series data by creating shorter representations of the original time series, such as the Discrete Fourier Transform [7], the Discrete Wavelet Transform (DWT) [8], the Piecewise Aggregate Approximation (PAA) [9], or the Symbolic Aggregation Approximation (SAX) [10].

3) **Interest point methods:** These methods focus on interest points (extrema) [11,12] rather than inspecting the entire time series. This leads to a

G. Bebis et al. (Eds.): ISVC 2013, Part I, LNCS 8033, pp. 280–289, 2013.

computational benefit (the number of extrema are much smaller than the number of data points on time series). The tradeoff here is, of course, that we might lose details of time series.

Different from the above approaches, TimeExplorer extracts features directly from the raw time series data. These features include some classic statistical summaries, such as means, standard deviations, and differences. Then we compute the dissimilarity of every two time series by using Euclidean distance in feature space.

Our contributions in this paper are:

- We have proposed a set of quantity measures of the raw time series data.
- We have developed a way to retrieve similar time series to a time series of interest-based Euclidean distance in feature space.
- We have proposed a method for filtering time series sharing common features.

The paper is structured as follows: We describe our time series features in the following section. Then we introduce our framework called TimeExplorer and illustrate it on real datasets. In Testing Section, we demonstrate the performance of our time series features on a commonly-used data mining dataset. Finally, we conclude the paper and present future explorations.

2 Time Series Features

The list of features that we describe in this section is not entirely new. Some features are classic statistical summaries, such as means and standard deviations. Some features have been studied individually by other researchers, such as sharp increases [13], mountains [14], and serial periodicities [15]. The objective of this section is to provide a complete list of features that are significant enough to summarize each time series. And we demonstrate this in the Testing Section.

Let $T = t_1, ..., t_n$ be a time series with n elements. Our features are classified into four groups depending on how many data points we consider at a time: one data point (raw data), two consecutive data points (differences), three consecutive data points, and longer subseries of data points.

2.1 One Data Point

These measures are computed based on raw data values where t_i is the data value at time i.

1) **Mean:** This is the mean value of an entire time series (with n data points).

$$Mean = \frac{\sum_{i=1}^{n} t_i}{n} \tag{1}$$

2) **Standard Deviation:** Standard Deviation (SD) shows how much variation exists around the Mean.

$$SD = \sqrt{\frac{\sum_{i=1}^{n} (t_i - Mean)^2}{n}} \tag{2}$$

2.2 Two Consecutive Data Points

These measure are computed based on the first differences of time series where $\mathit{difference}_i = t_i - t_{i-1}$. Differences are divided into increases (positive differences) and decreases (negative differences).

3) **Mean increase:** This is the average of increases where $n_{increase}$ is the number of increases over the time series.

$$Mean_{increase} = \frac{\sum_{i=2}^{n} increase_i}{n_{increase}} \tag{3}$$

4) **Mean decrease:** This is the average of decreases where $n_{decrease}$ is the number of decreases over the time series.

$$Mean_{decrease} = \frac{\sum_{i=2}^{n} decrease_i}{n_{decrease}} \tag{4}$$

5) **Max increase:** This is the maximum increase over the time series.

$$Max_{increase} = \max_{i=2}^{n} (increase_i) \tag{5}$$

6) **Max decrease:** This is the maximum decrease over the time series.

$$Max_{decrease} = \max_{i=2}^{n} (decrease_i) \tag{6}$$

7) **Standard Deviation differences:** Standard Deviation differences ($SD_{\mathit{difference}}$) shows how much variation exists from the $Mean_{\mathit{difference}}$.

$$SD_{\mathit{difference}} = \sqrt{\frac{\sum_{i=2}^{n} (\mathit{difference}_i - Mean_{\mathit{difference}})^2}{n-1}} \tag{7}$$

where

$$Mean_{\mathit{difference}} = \frac{\sum_{i=2}^{n} \mathit{difference}_i}{n-1} \tag{8}$$

2.3 Three Consecutive Data Points

These measures are computed based on three data points at a time. Specifically, we consider only two configurations: Mountain ($t_i > t_{i-1}$ and $t_i > t_{i+1}$) and Valley ($t_i < t_{i-1}$ and $t_i < t_{i+1}$). In the other words, a Mountain consists of one increase followed by one decrease. A Valley consists of one decrease followed by one increase. The Mountain and Valley at a data point i is the sum of the two slopes.

$$Mountain_i = |\mathit{difference}_{i-1}| + |\mathit{difference}_{i+1}| \tag{9}$$

$$Valley_i = |\mathit{difference}_{i-1}| + |\mathit{difference}_{i+1}| \tag{10}$$

8) **Max Mountain:** This is the maximum Mountain over the time series.

$$Max_{mountain} = \max_{i=2}^{n-1} (Mountain_i) \tag{11}$$

9) **Max Valley:** This is the maximum Valley over the time series.

$$Max_{valley} = \max_{i=2}^{n-1} (Valley_i) \tag{12}$$

2.4 Subseries

Let m be the length of a subseries in T. For example, m might be 7 for daily series or m might be 12 for monthly data. The following features measure how well two subseries match each other.

10) **Repeated:** The Repeated measure is the sum of differences of a subseries compared to the disjoint previous one.

$$Repeated = \sum_{i=m+1}^{n} |t_i - t_{i-m}| \qquad (13)$$

11) **Periodic:** The Repeated measure is high in a nearly single-valued time series. The Periodic measure looks for not only the repeated pattern but also a lot of variation in each subseries.

$$Periodic = Repeated * SD_{difference} \qquad (14)$$

All measures are normalized so that they receive the values from 0 to 1.

3 TimeExplorer Components

This section explains our approach in detail. Figure 1 shows a schematic overview:

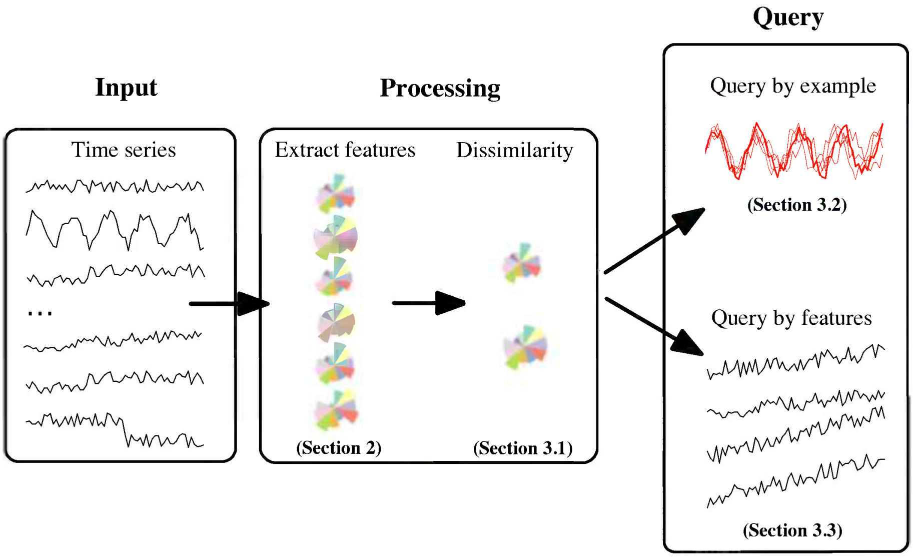

Fig. 1. Schematic overview of TimeExplorer

1. **Processing:** Our approach computes eleven features of each time series in the input data. Then, we compute the dissimilarity of each pair of time series based on the feature space (see Section 3.1).
2. **Query:** Users can select a time series from data overview graph to see all similar series or query time series by their features.

Information visualization systems should allow one to perform analysis tasks that largely capture people's activities while employing information visualization tools for understanding data [16]. The TimeExplorer implements three basic analysis tasks:

– Brushing: select a time series from the overview graph to retrieve all similar time series in the database (see Section 3.2).
– Sorting: sort time series based on their relevance to the selected time series (see Section 3.2).
– Filtering: find time series satisfying filtering conditions on their feature space (see Section 3.3).

3.1 Dissimilarity of Two Time Series

TimeExplorer offers several methods for discovering similar patterns in the time series. The dissimilarity of two time series (S and P) is computed by the following equation:

$$Dissimilarity(S, P) = \sqrt{\sum_{i=1}^{11} W_i (S_i - P_i)^2} \tag{15}$$

where S and P are two arrays of eleven features of the two time series and W_i is the weight of each feature (it is user input). TimeExplorer allows users to set weights for different features depending on data/applications and what they are looking for. In this paper, we set the same weights for all features.

The most obvious benefit of our parameterization is to reduce the complexity of searching for similar time series from $O(n)$ to $O(1)$ where n is the number of data points. That is, if we can characterize a time series with eleven features (mean, standard deviation, differences, periodicities, etc.), then we can make comparisons directly on these measures (instead of point to point comparisons). The tradeoff here is, of course, that we might lose details of time series. A way we ameliorate this problem is to provide selection tools to switch easily between different features. Our display changes almost instantly when a different features is selected for analysis. This feature allows an analyst to focus on a particular aspect of time series without excluding other possibilities.

3.2 Query by Example

We can select a time series by inputting its name in the search box, or by brushing an item from the overview graph. Figure 2 shows an example of querying similar unemployment situations in different cities in the US unemployment data. The data comprise monthly unemployment rates for 1772 cities (with population over 25,000 people) over 23 years from 1990 to 2012. The data were retrieved from http: //www.bls.gov/.

The top panel in Figure 2 shows the overview graph (1772 unemployment series graphed in a single display). The series selected by brushing is highlighted in red. The second panel displays only the selected time series and the top 5 similar series. We use colors to differentiate them. The more similar a time series is to the selected time series, the closer its color is to red on the rainbow scale. As we can see, all 6 displayed series are seasonal and follow each other nicely. The bottom panel displays the details of these series, including city names, their signatures, and their dissimilarities to the selected series. In particular, the query series is on the left, the top five series are ordered by similarity (the similarity decreases from left to right). The city names are colored using the encodings of the line graph in the second panel. Notice that we use the Nightingale Rose chart to display each time series signature. We display the feature names on the rose of the selected time series. The orientation and color of different features helps in comparing different signatures visually. Colors in the Nightingale Rose chart have been selected by Color Brewer at `http://colorbrewer.org/`.

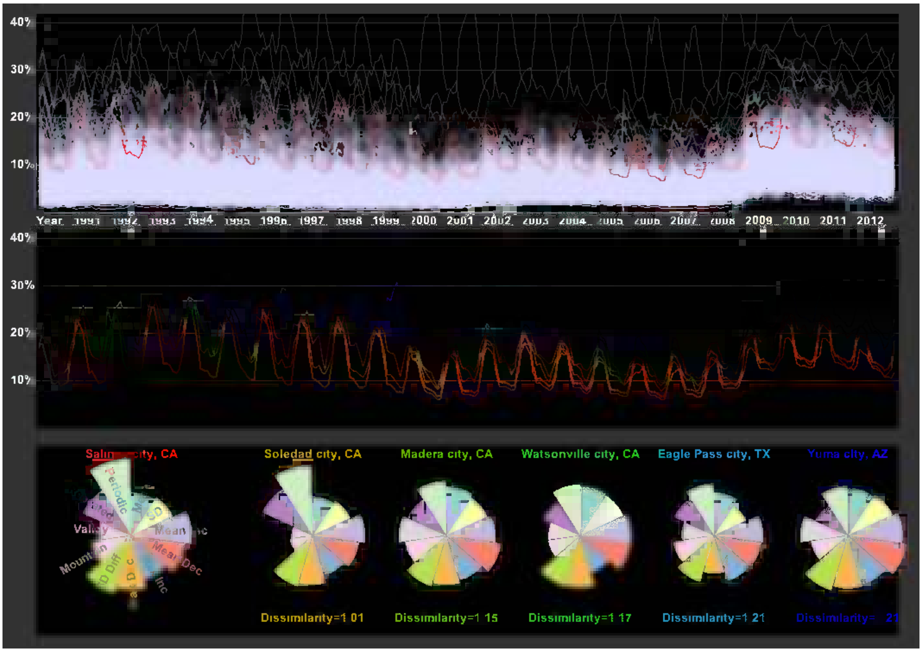

Fig. 2. US unemployment data: cities have similar periodic pattern as Salinas city, CA

3.3 Query by Features

We use a Nightingale Rose controller to filter time series features. To make our controller, we divide a circle into eleven sectors (associated with eleven time series features). We then divide each sector into three selection areas: outer, middle, and inner. Figure 3 shows an example of filtering standard deviation. By selecting the inner area (numbered 1 in the figure), we can control the lower bound of the range slider for standard deviation. By selecting the middle area (numbered 2), we can move both ends of the range slider at the same time. By selecting the outer area (numbered 3), we can control the upper bound of the range slider. We can select an area by mouse click and control an area by mouse scroll. The filtering condition of Figure 3 is $0.45 \leq SD \leq 0.75$.

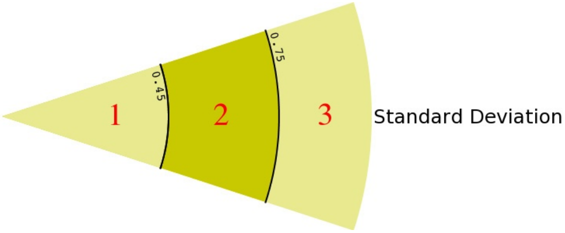

Fig. 3. Filtering standard deviation on a pie of Rose Controller

Figure 4 shows the top ten time series with the highest mountains in the Stock data. The data set contains 52 weekly stock prices for 1430 stocks. We obtain this results by setting the filtering condition: $Mountain \geq 0.8$. The Mountain feature deceases from top to down, and from left to right. The Nightingale Rose controller is depicted on the top of Figure 4.

4 Testing

In this section, we use the Synthetic Control Chart dataset [17] for testing. This dataset contains 600 synthetically generated control charts (60 data points on each time series). There are six different classes of control charts: Normal, Cyclic, Increasing trend, Decreasing trend, Upward shift, and Downward shift.

Testing approach:

1) We first extract eleven features from each chart.

2) Based on this feature space, we can compute a dissimilarity for each pair of time series using Euclidean distance.

3) We use leave-one-out cross-validation. This involves using a single chart from the original data as the validation data, and the remaining charts as the training data. This is repeated such that each chart in the data is used once as the validation data.

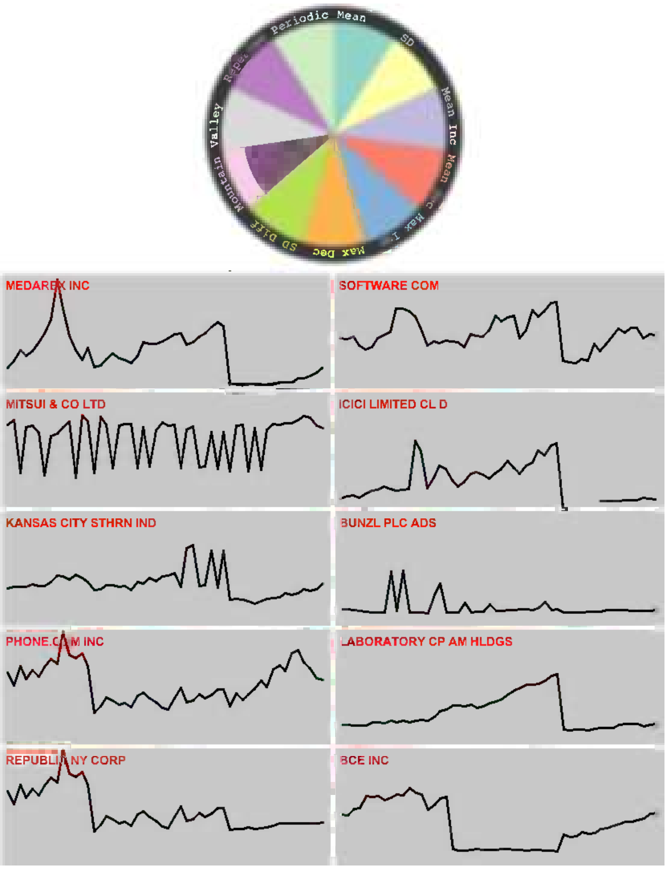

Fig. 4. The top 10 time series with the highest mountains in the Stock data

4) At every iteration, we use k-nearest neighbors (k-NN) to classify the input chart (validation data).

5) The classification obtained from Step 4 is then compared to the real class of the input chart to validate the result.

Table 1 shows our testing results on different nearest neighbors (k):

Table 1. Percentage of correct predictions of our approach

k-NN	k=1	k=3	k=5	k=7	k=9	k=11
Accuracy	93.8%	95.3%	95%	95.2%	95%	94.3%

Figure 5 shows the leave-one-out cross-validation results by 3-NN. The time series are colored by the true classification. Our classification results are displayed in six boxes in Figure 5. The time series with different colors from the color of

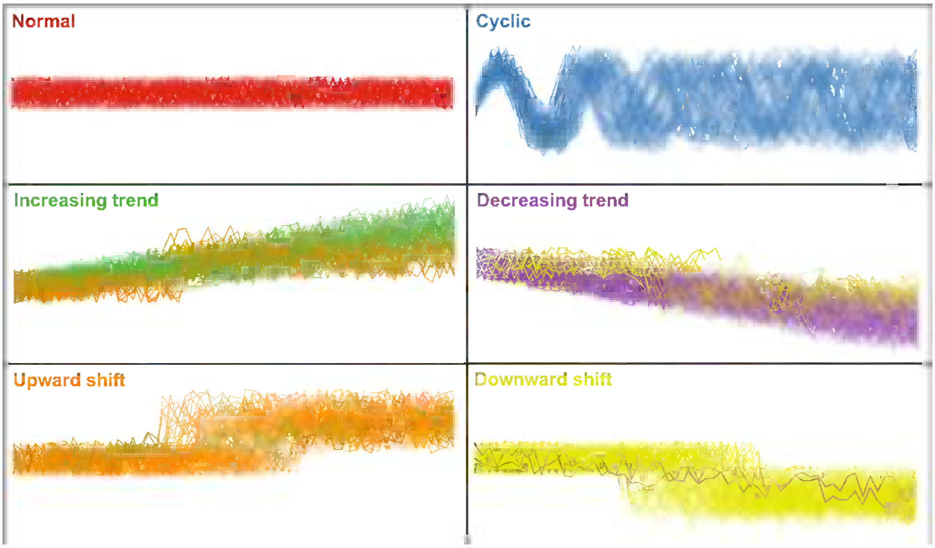

Fig. 5. Leave-one-out cross-validation results for the Synthetic Control Chart dataset

the boxes (group name) are wrong classifications. As depicted, most of the errors are Upward shift classified as Increasing trend and Downward shift classified as Decreasing trend.

5 Conclusion

In this paper, we propose a set of eleven features to compute similarity of time series. These features includes classic statistical summaries, such as means, standard deviations, and differences. We demonstrate the performance of these features through popular economic databases (the US unemployment data and the Stock data), as well as a synthetic dataset (Control Chart time series). The benefit of our approach is the compression we achieve by collapsing similarity searches from $O(n)$ to $O(1)$ through the use of eleven features where n is the number of data points in the time series. This provides TimeExplorer with the scalability to handle huge datasets with thousands of time series.

Finally, we plan to investigate the use of TimeExplorer on larger databases and real-time time series to assess the gains we claim for its performance. In addition, we expect to investigate the use of these features to cluster time series. This is useful for visualizing datasets with thousands of time series when the regular line graph becomes too cluttered to reveal salient trends or patterns. A summary line graph that shows only clusters of time series provides an comprehensive overview of huge time series database before further explorations.

References

1. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, SIGMOD 1994, pp. 419–429. ACM, New York (1994)
2. Li, L., Prakash, B.A.: Time series clustering: Complex is simpler! In: Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA, USA (2011)
3. Marteau, P.F.: Time warp edit distance with stiffness adjustment for time series matching. IEEE Trans. Pattern Anal. Mach. Intell. 31, 306–318 (2009)
4. Keogh, E., Lonardi, S., Chiu, B.Y.C.: Finding surprising patterns in a time series database in linear time and space. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002, pp. 550–556. ACM, New York (2002)
5. Rath, T.M., Manmatha, R.: Word image matching using dynamic time warping. In: CVPR (2), pp. 521–527 (2003)
6. Vlachos, M., Gunopoulos, D., Kollios, G.: Discovering similar multidimensional trajectories. In: Proceedings of the 18th International Conference on Data Engineering, ICDE 2002, p. 673. IEEE Computer Society, Washington, DC (2002)
7. Agrawal, R., Faloutsos, C., Swami, A.N.: Efficient similarity search in sequence databases. In: Lomet, D.B. (ed.) FODO 1993. LNCS, vol. 730, pp. 69–84. Springer, Heidelberg (1993)
8. Chan, K.P., Fu, A.C.: Efficient time series matching by wavelets. In: Proceedings of the 15th International Conference on Data Engineering, pp. 126–133 (1999)
9. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. In: Knowledge and Information Systems, vol. 3, pp. 263–286 (2001)
10. Lin, J., Keogh, E., Wei, L., Lonardi, S.: Experiencing sax: a novel symbolic representation of time series. Data Mining and Knowledge Discovery 15, 107–144 (2007)
11. Perng, C.S., Wang, H., Zhang, S., Parker, D.: Landmarks: a new model for similarity-based pattern querying in time series databases. In: Proceedings of the 16th International Conference on Data Engineering, pp. 33–42 (2000)
12. Vemulapalli, P., Monga, V., Brennan, S.: Optimally robust extrema filters for time series data. In: American Control Conference (ACC), pp. 2189–2195 (2012)
13. Shmueli, G., Jank, W., Aris, A., Plaisant, C., Shneiderman, B.: Exploring auction databases through interactive visualization. Decision Support Systems 42, 1521–1538 (2006)
14. Buono, P., Aris, A., Plaisant, C., Khella, A., Shneiderman, B.: Interactive pattern search in time series. In: Proc. SPIE, vol. 5669, pp. 175–186 (2005)
15. Carlis, J.V., Konstan, J.A.: Interactive visualization of serial periodic data. In: Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology, UIST 1998, pp. 29–38. ACM, New York (1998)
16. Amar, R., Eagan, J., Stasko, J.: Low-level components of analytic activity in information visualization. In: Proc. of the IEEE Symposium on Information Visualization, pp. 15–24 (2005)
17. Asuncion, A., Newman, D.: UCI machine learning repository (2007), http://archive.ics.uci.edu/ml/datasets/Synthetic+Control+Chart+Time+Series