Rule discovery from time series

Gautam Das
Department of
Mathematical Sciences
University of Memphis
Memphis, TN 38152
USA
dasg@msci.memphis.edu

King-Ip Lin
Department of
Mathematical Sciences
University of Memphis
Memphis, TN 38152
USA
linki@msci.memphis.edu

Heikki Mannila
Department of
Computer Science
University of Helsinki
P.O. Box 26, FIN-00014 Helsinki,
Finland
mannila@cs.helsinki.fi

Gopal Renganathan
Autozone Inc.,
123 So. Front St.,
Memphis, TN 38103
USA
Gopal.Renganathan@crv.autozone.com

Padhraic Smyth
Department of
Information and Computer Science
University of California at Irvine
CA 92697-3425, USA
smyth@ics.uci.edu

Abstract

We consider the problem of finding rules relating patterns in a time series to other patterns in that series, or patterns in one series to patterns in another series. A simple example is a rule such as "a period of low telephone call activity is usually followed by a sharp rise in call volume". Examples of rules relating two or more time series are "if the Microsoft stock price goes up and Intel falls, then IBM goes up the next day," and "if Microsoft goes up strongly for one day, then declines strongly on the next day, and on the same days Intel stays about level, then IBM stays about level." Our emphasis is in the discovery of *local* patterns in multivariate time series, in contrast to traditional time series analysis which largely focuses on *global* models. Thus, we search for rules whose conditions refer to patterns in time series. However, we do not want to define beforehand which patterns are to be used; rather, we want the patterns to be formed from the data in the context of rule discovery. We describe adaptive methods for finding rules of the above type from time-series data. The methods are based on discretizing the sequence by methods resembling vector quantization. We first form subsequences by sliding a window through the time series, and then cluster these subsequences by using a suitable measure of time-series similarity. The discretized version of the time series is obtained by taking the cluster identifiers corresponding to the subsequence. Once the time-series is discretized, we use simple rule finding methods to obtain rules from the sequence. We present empirical results on the behavior of the method.

Keywords: time series, rules, clustering, discretization, vector quantization

Introduction

Time series data occurs frequently in business applications and in science. Well-known examples include daily stock prices at the New York Stock Exchange, hourly volumes of telephone calls between the United States and Europe, and daily sea-surface temperature readings in the Pacific. There has been a lot of interest into querying time series on the basis of similarity (see, e.g., (Agrawal, Faloutsos, & Swami 1993; Shatkay & Zdonik 1996; Agrawal *et al.* 1995; Rafiei & Mendelzon 1997; Yazdani & Ozsoyoglu 1996)).

In this paper we are interested in finding rules relating the behavior of patterns within a sequence over time, or the relationship between two or more sequences over time. An example would be a rule such as "a period of gradual increase in the value of sea-surface temperature over the South Pacific is typically followed by sharp increase in precipitation over the Western United States." Rules typically assume an underlying symbolic (or propositional) representation, whereas our specific interest here is in real-valued time series. The novel contribution in this paper is in the extraction of a discrete data-driven pattern representation from the time-series, and then using this representation as the basis for exploratory rule induction.

A time series can be converted into a discrete representation by first forming subsequences (using a sliding window) and then clustering these subsequences using a suitable measure of pattern similarity. The discretized version of the time series is obtained by using the cluster identifiers corresponding to the subsequence, in a manner which is similar to the data compression technique of vector quantization. Rule-finding algorithms (such as the "episode rule" methods) can then be used directly on the discretized sequence to uncover rules relating temporal patterns. Thus our rule discovery method aims at finding *local relationships* from the series, in the spirit of association rules, sequential patterns, or episode rules (Agrawal, Imielinski, & Swami 1993; Agrawal & Srikant 1995; Mannila, Toivonen, & Verkamo 1997). Unlike traditional time series modeling, we do not seek a global model for the time series, instead searching for local patterns in a relatively non-parametric manner.

As an example of the results of the method, from the daily closing share prices of 10 database companies traded on the NASDAQ we can find several informative rules. For example consider the rule "if a stock price follows the pattern s18 in figure 2 (a), then within 20 days it will exhibit th epattern s4 shown in the same figure.". This can be interpreted as a pattern of decline. In addition, we can find rules relating the behavior of the stocks of individual companies. For instance, our method discovered a number of rules relating two database companies. Namely, if the stock of one company has approximately the pattern 15 shown in Figure 3 (a), then the stock of the other company will exhibit similar behavior within a month. Examining the data, we find that the two companies are both object-oriented database companies.

Time-series discretization by clustering

Basic method

Our method for discretizing a time-series by clustering windows is as follows. Suppose we are given a sequence s and a window width w. Given $s = (x_1, \ldots, x_n)$, a window of width w on s is a contiguous subsequence $(x_i, \ldots, x_{i+w-1})$. We form from s all windows (subsequences) $s_1, \ldots, s_{n-w+1}$ of width w, where $s_i = (x_i, \ldots, x_{i+w-1})$. Denote $W(s) = \{s_i \mid i = 1, \ldots, n-w+1\}$.

Assume we have a distance $d(s_i, s_j)$ between any two subsequences s_i and s_j of width w. These distances can be used to cluster the set of all subsequences $W(s)$ into sets $C_1, \ldots, C_k$. For each cluster C_h we introduce a symbol a_h, and the discretized version $D(s)$ of the sequence s will be over the alphabet $\Sigma = \{a_1, \ldots, a_k\}$. The sequence $D(s)$ is obtained by looking for each subsequence s_i the cluster $C_{j(i)}$ such that $s_i \in C_{j(i)}$, and using the corresponding symbol $a_{j(i)}$. Thus

$$D(s) = a_{j(1)}, a_{j(2)}, \ldots, a_{j(n-w+1)}$$

Essentially, each symbol a_h represents a primitive "shape", and we are interested in discovering rules that involve patterns composed of these basic shapes. An example is shown in Figure 1.

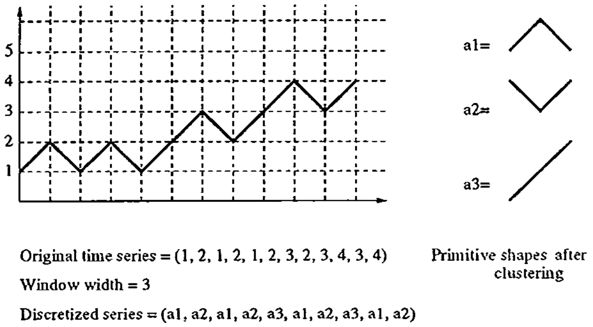


Figure 1: Example of rules based on basic shapes

The discretization process described above depends on the choice of w, on the choice of the time-series distance function, and on the type of clustering algorithm used.

The basic property of this method is that the alphabet is derived from the data; it is not provided by a domain expert (although our methods can be generalized to allow such inputs). Using data-derived patterns as "primitives" in a more abstract representation of a signal is not a new idea. Essentially the same method is used in the well-known vector quantization (VQ) method of data compression (see, e.g., (Gersho & Gray 1992)). VQ is based on the notion of replacing local windows (of size w) (of signals or images) by pattern centroids determined by an algorithm quite similar to k-means clustering. For data compression, only the indices of the centroids need to be transmitted, permitting signal compression at the cost of some fidelity. Thus, one can view the methods proposed in this paper as the application of VQ (combined with rule induction) to *signal understanding* rather than the more typical VQ task of *signal compression*.

We are advocating a very general approach towards rule discovery in time series databases. The user has the choice of a variety of methods to employ. It cannot be overemphasized that the rule discovery process is an iterative activity. Each time the system discovers certain rules, domain experts should analyze and interpret the resulting rules. The discovery algorithms should be run several times with different parameter settings. Different runs provide different views of the underlying dataset. For example, a small w may produce rules that describe short term trends, while a large w may produce rules that give a more global view of the dataset. One can also run the method at different scales (by subsampling the data) allowing for multi-resolution data exploration (these options are not explored in this paper due to space constraints).

The running time of the algorithm is dominated by the time needed to cluster the $O(n)$ subsequences resulting from the windowing method. It is also possible to consider only every vth window for some integer v.

In the next subsections we describe the time-series similarity notions used and the clustering methods employed.

Notions of time-series similarity

To cluster the set $W(s)$ we need a distance notion for time-series of length w. There are several possibilities, and the specific choice for any given application should depend on the specific manner the application environment is generating the observed time series. In this section we describe some possible distance measures and discuss their use in rule discovery.

The simplest possibility is to treat the subsequences of length w as elements of R^w and use the Euclidean distance (i.e., the L_2 metric). That is, for $\bar{x} = (x_1, \ldots, x_w)$ and $\bar{y} = (y_1, \ldots, y_w)$ we define $d(\bar{x}, \bar{y}) = (\sum_i (x_i - y_i)^2)^{1/2}$ as the metric in clustering. Other alternative metric includes the general L_p metrics defined by $L_p(\bar{x}, \bar{y}) = (\Sigma_i (x_i - y_i)^{\frac{1}{p}})$ for $p \geq 1$ and $L_\infty = \max_i \mid x_i - y_i \mid$.

For many applications one would like to see the shape of the subsequence as the main factor in distance determination. Thus, two subsequences may have essentially the same shape, although they may differ in their amplitudes and baselines. One way of achieving this is by *normalizing* the subsequences and then using the L_2 metric on the normalized subsequences. Denoting the normalized version of sequence $\bar{x}$ by $\eta(\bar{x})$; we define the distance between $\bar{x}$ and $\bar{y}$ by $d(\bar{x}, \bar{y}) = L_2(\eta(\bar{x}), \eta(\bar{y}))$.

One possible normalization is $\eta(\bar{x})_i = x_i - E\bar{x}$, (where $E\bar{x}$ is the mean of the value of the sequence) which makes the mean of the sequence to be 0. Another possibility is $\eta'(\bar{x})_i = (x_i - E\bar{x})/D\bar{x}$, (where $D\bar{x}$ is the standard deviation of the sequence) forcing the mean to be 0 and the variance 1.

Recently, more sophisticated time series distance measures have been investigated, such as the *dynamic time warping* (Berndt & Clifford 1994) measure, the *longest common subsequence* measure (Das, Gunopulos, & Mannila 1997; Bollobás *et al.* 1997), and various probabilistic distance measures (Keogh & Smyth 1997). Due to space limitations we omit the details of their use but note that the results below can be easily generalized to handle any such distance measures.

Clustering methods

The first step in the discretization process is the clustering. Recall that w is one of the parameters to the system; it is used to define the set $W(s)$. In principle, any clustering algorithms can be used to cluster the subsequences in $W(s)$; see (Jain & Dubes 1988; Kaufman & Rousseauw 1990) for overviews. We have experimented with the following two methods.

The first method is a greedy method for producing clusters with at most a given diameter. Treat each subsequence in $W(s)$ as a point in R^w, and let us use the L_2 metric as distance between the points. Let a small constant $d > 0$ be another parameter to the clustering algorithm.

For each point p in $W(s)$, the method finds the cluster center q such that $d(p, q)$ is minimum. If $d(p, q) < d$ then p is added to the cluster whose center is q, otherwise a new cluster with center p is formed. Once the algorithm has examined all the points in $W(s)$, let the cluster centers be $q_1, \ldots, q_k$. It is easy to see that the distance between two cluster centers is at least d, while the radius of each cluster is at most d.

We also used the traditional k-means algorithm, where cluster centers for k clusters are initially chosen at random among the points of $W(s)$. In each iteration, each subsequence of $W(s)$ is assigned to the cluster whose center is nearest to it. After this, for each cluster its center is recomputed as the pointwise average of the sequences contained in the cluster. The iterations are continued until the process converged. This method is widely used; one disadvantage is that the number of clusters has to be known in advance.

Once the clustering is complete, each cluster center represents a basic "shape." The alphabet of cluster centers is then used to encode the original time series, as discussed above.

Remarks on the choice of parameters

Note that the entire discretization process depends on several parameters. Among them are the window width (w), the maximal cluster diameter (d) or the number (k) of clusters. Other parameters may also be used in the preprocessing stages (like window movement v). How do we know whether a certain combination of parameters produces a "good" discretization?

The choice of the window width w (and window movement v) depend on the time scale the user is interested in. Thus, no specific guidelines can be provided in the general case, rather the user must choose his/her values based on their own particular bias and application considerations. One of the useful aspects of our approach is that we can look at the sequence using several granularities.

For the cluster diameter d or the number of clusters k, there are some intuitive yardsticks that can be used. The eventual goal is the discovery of interesting, interpretable, and useful rules. Too many clusters will not help in this: it will be almost impossible to associate understandable interpretations to so many basic abstract "shapes." Likewise, too few clusters will not help, as each cluster contains subsequences that are too far away from each other.

A simpler strategy is to ignore the issue of whether the discretization is good or not until *after* the rules have been discovered. A good discretization is one that produces interesting rules. One (but not the only) criterion for interestingness is estimated informativeness, i.e., whether the rule gives additional information about the sequences. We can assign a measure of informativeness to the discovered rules using the J-measure (Smyth & Goodman 1991; 1992).

One can also run the method for several choices of the parameters and let the user browse the different rule sets. The running time of the method is small enough so that this is feasible. An extension of this idea is to have the algorithm search over different value of w and d and to return the most informative rules over these values.

Rule discovery from discretized sequences

Rule format

In this section we outline algorithms that discover simple rules from a set of discretized sequences. The simplest rule format is:

if A occurs, then B occurs within time T.

Here A and B are basic shapes, i.e., the are letters from the alphabet produced by the discretization. We write the above rule as $A \overset{T}{\Rightarrow} B$.

Given a sequence $D(S) = (a_1, a_2, \ldots, a_n)$, the *frequency* $F(A)$ of the rule A is the number of occurrences of $A \in D(S)$, and the relative frequency $f(A)$ of A is $F(A)/n$. The *confidence* $c(A \overset{T}{\Rightarrow} B)$ of the rule $A \overset{T}{\Rightarrow} B$ is the fraction of occurrences of A that are followed by a B within T units, i.e.,[1]

$$c(A \overset{T}{\Rightarrow} B) = \frac{F(A,B,T)}{F(A)},$$

where

$$F(A,B,T) = \mid \{i \mid a_i = A \wedge B \in \{a_{i+1}, \ldots, a_{i+T-1}\}\} \mid$$

is the number of occurrences of A that are followed by a B within T.

A slight modification is, however, often useful. Recall that the two consecutive letters c_i and c_{i+1} in the discretized sequence $D(s)$ come from two windows of width w which have an overlap of $w - v$. Thus, consecutive letters are strongly correlated, and we tend to get rules with high confidence that actually are just a by-product of the discretization method. Therefore it typically makes sense to define rule confidence by

$$F(A,B,T) = \\ \mid \{i \mid a_i = A \wedge B \in \{a_{i+w+1}, \ldots, a_{i+w+T-1}\}\} \mid$$

i.e., to count only occurrences of B that occur after w units of time.

Computing the frequencies and confidences of such rules is easy, by a simple pass through the sequence. The number of possible rules is mk^2, where k is the number of letters in the alphabet and m is the number of different possibilities for T.

Informative Rules

The above method produces lots of rules, with varying confidences. For interactive knowledge discovery, a good strategy is to allow the user to browse through rule sets and provide tools for the selection of interesting rules (Kloesgen 1995; Klemettinen *et al.* 1994; Brin, Motwani, & Silverstein 1997). Nonetheless, no single significance criterion can probably suffice to select the most valuable rules. Still, the user needs some guidance in determining which rules have a confidence that differs substantially from the expected.

There are a variety of metrics which can be used to rank rules (e.g., see (Piatetsky-Shapiro 1991) for a general overview of such methods). Here we use the J-measure for rule-ranking (Smyth & Goodman 1991; 1992) defined as:

$$J(B_T; A) = p(A) * \Big(p(B_T|A) \log(\frac{p(B_T|A)}{p(B_T)}) \\ +(1 - p(B_T|A)) \log(\frac{1 - p(B_T|A)}{1 - p(B_T)})\Big)$$

where, in the context of sequence rules, $p(A)$ is the probability of symbol A occurring at a random location in the sequence, $p(B_T)$ is the probability of at least one B occurring in a randomly chosen window of duration t and $p(B_T|A)$ is the probability of at least one B occurring in a randomly chosen window of duration T given that the window is immediately preceded by an A. Intuitively, the first term in the J-measure, namely $p(A)$, is a bias towards rules which occur more frequently. The second term is well-known as the cross-entropy, namely the information gained (or degree of surprise) in going from a prior probability $p(B_T)$ to a posterior probability $p(B_T|A)$. As shown in (Smyth & Goodman 1992) the product of the two terms (the J-measure above) has unique properties as a rule information measure and is in a certain sense a special case of Shannon's mutual information. From a practical viewpoint, the measure provides a useful and sound method for ranking rules in a manner which trades-off rule frequency and rule-confidence. Note that in estimating the probabilities in the equation for the J-measure (above) it is helpful to use simple maximum a posteriori estimates ("smoothed" counts) rather than maximum likelihood estimates (see (Smyth & Goodman 1991) for further discussion of this point).

Extensions

The basic method can be extended in various ways. We describe briefly some possibilities.

Multiple time series It is straightforward to extend the previous framework for rules between two series. Given m sequences $D(S_h) = (c_{h1}, c_{h2}, \ldots, c_{hn})$ for $h = 1, \ldots, m$, a rule still has the form of $A \overset{T}{\Rightarrow} B$, while A and B might come from different discretizations. The

[1] Note that this differs from the usage of frequency or support for association rules (Agrawal, Imielinski, & Swami 1993; Agrawal *et al.* 1996), where frequency is defined as the fraction of objects that satisfy the left and right hand sides of the rule.

definition of frequency, confidence and significance is the same as the previous definitions.

Extending the rule format The rule format above can be extended to include rules of the form

> if A_1 and A_2 and ... and A_h occur within V units of time, then B occurs within time T,

denoted $A_1 \wedge \cdots \wedge A_h \overset{V,T}{\Rightarrow} B$. The frequency of such a rule can be defined as the number of occurrences of A_1 that are followed by A_2 etc. within time V. Rules of this type have been studied under the name *sequential patterns* (Agrawal & Srikant 1995) and *episode rules* (Mannila, Toivonen, & Verkamo 1997), and the algorithms developed there can be used in this context also.

The problem with this extension is that the number of potential rules grows quickly. For rules with h letters on the left hand side we have to prune rules on the basis of frequency. That is, in order for the rule $A_1 \wedge \cdots \wedge A_h \overset{V,T}{\Rightarrow} B$ to be considered, the frequency of the rule has to exceed a given threshold. This technique stems from association rule algorithms (Agrawal, Imielinski, & Swami 1993; Agrawal *et al.* 1996), and it is very efficient in pruning the search space; the drawback is that rules with high significance but low frequency might go undetected (Brin, Motwani, & Silverstein 1997). Another possibility here (which we did not experiment with) is to use the branch-and-bound properties of the J-measure to prune the search space, as in the ITRULE algorithm of (Smyth & Goodman 1992).

Experimental results

We used three different data sets in our experiments:

1. Stock data: daily closing prices of 10 database companies traded on the Nasdaq stock market for the past 19 months. Each sequence is of length 410.
2. Telecommunications data: traffic volumes on 34 lines in the Helsinki metropolitan area. The volume is recorded every 15 minutes, and the series have length 478 (approximately 5 days).
3. Paleoecological data: abundances of 36 different taxa of diatoms in sediment at the bottom of a lake in northern Finland, at 147 different depths. That is, there are 36 different series, each having length 147.

Our experiments focused on finding out whether the method discovers interesting rules from the sequences, and whether the method is robust enough so that small changes in the values of the parameters do not change the results drastically.

For each of the data sets, we experimented with several different window widths w, rule time lengths T, cluster diameter d, and the number of clusters k. For each experiment, the resulting rules were ranked using the J-measure. Due to the lack of space, we present only a small subset of the results, concentrating on the stock data set.

Simple rule discovery We first set up a minimum threshold of 1% frequency and 50% confidence. Any rules that do not meet this criteria were discarded. After that we use the J-measure to compare the quality of the rules. The top scoring rules are listed in the following table. Figure 2 shows the centers of the clusters for each of the rules.

w	d	Rule	Sup. (%)	Conf. (%)	J-mea.	Fig.
13	3.5	$18 \overset{20}{\Rightarrow} 4$	2.8	59.6	0.0037	2(a)
15	4.0	$37 \overset{20}{\Rightarrow} 42$	1.3	57.37	0.0087	2(b)
15	4.5	$11 \overset{20}{\Rightarrow} 9$	3.5	66.7	0.0031	2(c)
30	5.5	$76 \overset{20}{\Rightarrow} 21$	1.2	57.3	0.0073	2(d)

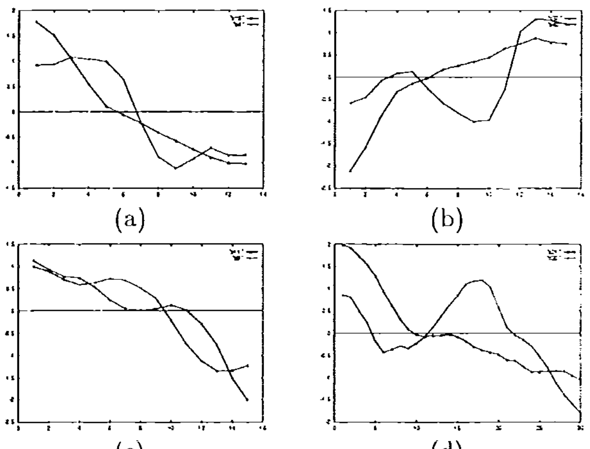
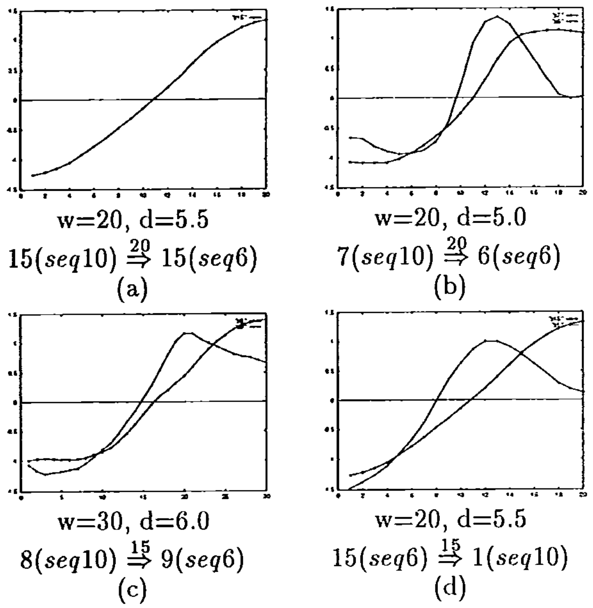

(a) (b) (c) (d)

Figure 2: Significant rules for stock data

An interpretation of the rule in figure 2 (a) is that a stock which follows a 2.5-week declining pattern of *s18* (sharper decrease and then leveling out), will likely incur a short sharp fall within 4 weeks before leveling out again (the shape of *s4*).

Rules for pairs of sequences We also compare individual sequences to detect applicable rules for each pair of sequences. In this case, we put a higher threshold on the minimum support to generate more meaningful rules. We found the top set of 488 rules (rules with J-measure > 0.03). Figure 3 shows some example rules that describe patterns found from the time series of stock prices of two object-oriented database companies. While these time series as a whole are not very similar, there is substantial evidence of links between the two series, as the local patterns demonstrate quite strongly.

Also we discovered that more than 27% of the rules relate 3 pairs of sequences, out of a total of 45 (i.e. 2%). This indicates that only a small set of sequences are closely related.



w=20, d=5.5
$15(seq10) \overset{20}{\Rightarrow} 15(seq6)$
(a)

w=20, d=5.0
$7(seq10) \overset{20}{\Rightarrow} 6(seq6)$
(b)

w=30, d=6.0
$8(seq10) \overset{15}{\Rightarrow} 9(seq6)$
(c)

w=20, d=5.5
$15(seq6) \overset{15}{\Rightarrow} 1(seq10)$
(d)

Figure 3: Significant rules for sequence 6 and 10

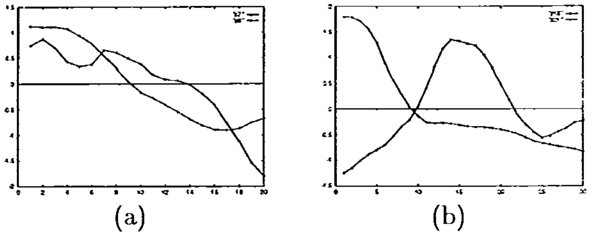
Robustness as a function of the clustering methodology We also investigated the impact of the clustering methodology on the results of the algorithm. We considered some of the rules in the previous section, and tried to find a similar rule using different values of d and/or w. The experimental results showed that as the clustering parameters are perturbed, the effect is to also induce slight changes in the rules which are discovered, i.e., the new rules which are discovered are in the neighborhood of the old rules discovered by the previous parameter settings, indicating that the method is reasonably robust.

Another important aspect of the clustering methodology is the clustering algorithm itself. In the previous experiments, clustering is carried out by using the K-means algorithm, which means that the algorithm will iterate until the clusters are stable. Experiment shows that the number of iterations ranges between 10 and 80 for the stock data. Obviously, for large data sets, reading the data multiple times is unrealistic. One alternative is to employ the greedy method of clustering mentioned previously.

Our experiments discovered many similar rules even using the greedy algorithm. A couple of significant rules are shown in following table and figure 4:

w	d	Rule	Sup. (%)	Conf. (%)	J-mea.	Fig.
20	5.5	$7 \overset{15}{\Rightarrow} 8$	8.3	73.0	0.0036	4(a)
30	5.5	$18 \overset{20}{\Rightarrow} 21$	1.3	62.7	0.0039	4(b)

The two rules in this figure corresponds to rules shown in figure 2 (c) and (d) respectively. Thus, empirically, we can speed up the clustering process without any significant change in the discovered rules. An interesting problem for further research is to more precisely characterize and quantify the trade-off between computational costs of the algorithm and the quality of the discovered rules.



(a) (b)

Figure 4: Significant rules for stock data (using only 2 iteration of K-means)

Discussion

Extracting rules directly from time-series data involves two coupled problems. First, since rules are inherently a symbolic representation, one must transform the low-level signal data into a more abstract symbolic alphabet. In this paper this is achieved by data-driven clustering of signal windows in a similar way to that used in VQ data compression.

The second problem is that of rule induction from symbolic sequences. Naturally, there is a trade-off between the quality of the abstraction and the quality of the rules induced using this abstraction. Parameters such as cluster window width, clustering methodology, number of clusters, and so forth, may well affect the types of rules which are induced. In this context it is important to keep in mind that the proposed technique is essentially intended as an *exploratory* method and thus, iterative and interactive application of the method coupled with human interpretation of the rules is likely to lead to the most useful results (rather than any fully automated approach). Our methods are first steps, and additional experimentation is needed to estimate the strengths and weaknesses of the method.

Clearly there are several directions for generalizing the concepts introduced here, such as alternative abstractions (rather than pattern centroids). For example, the hierarchical piecewise linear representation introduced in (Keogh & Smyth 1997) may provide a computationally efficient way to increase the expressive power of the underlying signal representation. The piecewise linear data structure implicitly handles variability in "warping" of signal structure (e.g., signal peaks which may be amplitude-scaled and/or stretched in time), a feature which is absent in the "fixed window" method described here. Furthermore, a hierarchical representation may provide a practical way to incorporate the notion of multi-resolution scale into the representation in a natural manner, allowing for rules which relate events at different scales in the signal structure.

Also generalizing the rule language is an interesting problem. for example allowing regular expressions over the patterns produced by the clustering (in the spirit of (Agrawal *et al.* 1995)).

References

Agrawal, R., and Srikant, R. 1995. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering (ICDE'95)*, 3 - 14.

Agrawal, R.; Psaila, G.; Wimmers, E. L.; and Zait, M. 1995. Querying shapes of histories. In *Proceedings of VLDB.*

Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.; and Verkamo, A. I. 1996. Fast discovery of association rules. In Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurusamy, R., eds., *Advances in Knowledge Discovery and Data Mining.* Menlo Park, CA: AAAI Press. 307 - 328.

Agrawal, R.; Faloutsos, C.; and Swami, A. 1993. Efficiency similarity search in sequence databases. In *Proceedings of the Conference on Foundations of Data Organization*, 22. IBM Almaden Research Center.

Agrawal, R.; Imielinski, T.; and Swami, A. 1993. Mining association rules between sets of items in large databases. In Buneman, P., and Jajodia, S., eds., *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, 207 - 216. Washington, D.C., USA: ACM.

Berndt, and Clifford. 1994. Using dynamic time warping to find patterns in time series. In *Proceedings of AAAI Workshop on Knowledge Discovery in Databases 1994*.

Bollobás, B.; Das, G.; Gunopulos, D.; and Mannila, H. 1997. Time-series similarity problems and well-separated geometric sets. In *13th Annual ACM Symposium on Computational Geometry*, 454–456.

Brin, S.; Motwani, R.; and Silverstein, C. 1997. Beyond market baskets: Generalizing association rules to correlations. In Peckman, J. M., ed., *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'97)*, 265 - 276. Tucson, AZ: ACM.

Das, G.; Gunopulos, D.; and Mannila, H. 1997. Finding similar time series. In *Principles of Knowledge Discovery and Data Mining (PKDD) 1997.*

Gersho, A., and Gray, R. M. 1992. *Vector Quantization and Signal Compression.* Boston: Kluwer Academic Publishers.

Jain, A. K., and Dubes, R. C. 1988. *Algorithms for Clustering Data.* Englewood Cliffs, NJ: Prentice-Hall.

Kaufman, L., and Rousseauw, P. J. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley and Sons.

Keogh, E., and Smyth, P. 1997. A probabilistic approach to fast pattern matching in time series databases. In Heckerman, D.; Mannila, H.; Pregibon, D.; and Uthurusamy, R., eds., *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, 24. AAAI Press.

Klemettinen, M.; Mannila, H.; Ronkainen, P.; Toivonen, H.; and Verkamo, A. I. 1994. Finding interesting rules from large sets of discovered association rules. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, 401 - 407. Gaithersburg, MD: ACM.

Kloesgen, W. 1995. Efficient discovery of interesting statements in databases. *Journal of Intelligent Information Systems* 4(1):53 - 69.

Mannila, H.; Toivonen, H.; and Verkamo, A. I. 1997. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery* 1(3):259 - 289.

Piatetsky-Shapiro, G. 1991. Discovery, analysis, and presentation of strong rules. In Piatetsky-Shapiro, G., and Frawley, W. J., eds., *Knowledge Discovery in Databases.* Menlo Park, CA: AAAI Press. 229 - 248.

Rafiei, D., and Mendelzon, A. 1997. Similarity-based queries for time series data. *SIGMOD Record (ACM Special Interest Group on Management of Data).*

Shatkay, H., and Zdonik, S. B. 1996. Approximate queries and representations for large data sequences. In *Proceedings of the 12th International Conference on Data Engineering*, 536–545. Washington - Brussels - Tokyo: IEEE Computer Society.

Smyth, P., and Goodman, R. M. 1991. Rule induction using information theory. In *Knowledge Discovery in Databases.* Cambridge: MA: The MIT Press. 159–176.

Smyth, P., and Goodman, R. M. 1992. An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Engineering* 4(4):301–316.

Yazdani, N., and Ozsoyoglu, Z. M. 1996. Sequence matching of images. In *Proceedings of 8th International Conference on Scientific and Statistical Database Management (SSDBM)*, 53 - 62.