THE INDIAN SOCIETY FOR HYDRAULICS
JOURNAL OF HYDRAULIC ENGINEERING

DATA MINING AND ITS APPLICATIONS FOR MODELLING RAINFALL EXTREMES

by

D. Nagesh Kumar[1], M.ISH and **C.T. Dhanya**[2]

ABSTRACT

Data mining is a new powerful technology which helps in extracting hidden predictive information (future trends and behaviours) from large databases and thus facilitating decision makers to make proactive, knowledge-driven decisions. In this paper, a brief overview of various data mining functionalities, and an extensive review of the works done on temporal data mining are discussed. Of the two frameworks of temporal data mining, one that of frequent episodes is discussed in detail by explicating the various algorithms developed so far. Also, a case study using one of the algorithms, Minimal Occurrences With Constraints And Time Lags (MOWCATL), for extracting the rules to explain the spatial and temporal variation for extreme events in India is discussed and the results are shown.

KEYWORDS: Data mining; Temporal data mining; Association rules; Frequent episodes; Indian summer monsoon rainfall; Extreme rainfall events; Climatic Indices.

INTRODUCTION

There has been an exponential increase in the amount of information or data being stored in electronic format in the last three decades, especially in the field of business operations. The availability of large amounts of storage spaces at low cost with the innovation of powerful computers led to the introduction of machine learning methods for knowledge representation in addition to the traditional statistical analysis of data. In spite of the skillfulness of the traditional database management systems in putting data in an easily accessible manner, these are not intended to analyze the data explicitly stored nor provide any further information by going beyond the data. These stored data have to be in some way converted to knowledge or information. This is where the concept of data mining or knowledge discovery process can bring aid to any enterprise.

1. Professor, Department of Civil Engineering, Indian Institute of Science, Bangalore 560012, India
2. Ph.D Scholar, Department of Civil Engineering, Indian Institute of Science, Bangalore 560012, India

Note : Written discussion of this paper will be open until 30th June 2009.

Data mining techniques have been mainly used by enterprises to manage their consumer relationships and also to decide upon the relationships between commodity price, availability of products, staff skills and customer details, frugality, economy. These endue them to determine the impact on corporate profits, marketing and also consumer satisfaction. The last decade has shown the extensive use of data mining techniques (especially association rules, clustering and decision trees) being applied to various engineering fields. Also, the scope of the term data mining is expanded to apply to any form of data analysis. Accordingly, there are numerous definitions of data mining of which some are mentioned below.

Data mining in simple terms refers to the extraction or mining of knowledge from large amounts of data. The analogy between the term data mining and the mining process is described in Clementine User Guide (ISL, 1994) by referring data mining to *"using a variety of techniques to identify nuggets of information or decision-making knowledge in bodies of data, and extracting these in such a way that they can be put to use in the areas such as decision support, prediction, forecasting and estimation. The data is often voluminous, but as it stands of low value as no direct use can be made of it; it is hidden information in the data that is useful"*.

Data mining is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data (Frawley et al., 1991). *Data mining is the search for relationships and global patterns that exist in large databases but are hidden among the vast amount of data. These relationships represent valuable knowledge about the database and, if the database is a faithful mirror, of the real world registered by the database* (Holshemier & Siebes, 1994).

The terms such as knowledge mining from data, knowledge extraction, data analysis and data dredging are many a time used as synonyms to data mining. Likewise, many people use the term Knowledge Discovery from Data (KDD) as a synonym to data mining. On the other hand, some others treat data mining as one of the steps in the process of knowledge discovery. The knowledge discovery process is shown diagrammatically in fig. 1 in which it is depicted as an iterative sequence consisting of seven steps (Han & Kamber, 2006):

i. Data cleaning: remove noise and inconsistent data and also unnecessary data which may slow down the queries.
ii. Data integration: data from multiple sources are integrated and reconfigured to a consistent format, if necessary.
iii. Data selection: data relevant for the analysis are retrieved from the database
iv. Data transformation: data are transformed into forms appropriate for mining
v. Data mining: A crucial step where intelligent methods are applied to extract patterns from the data

vi. Pattern evaluation: Among all patterns extracted using the data mining techniques, the truly interesting patterns are identified based on some interestingness measures

vii. Knowledge presentation: patterns identified are made easily accessible to human decision making, employing data visualization and knowledge representation techniques

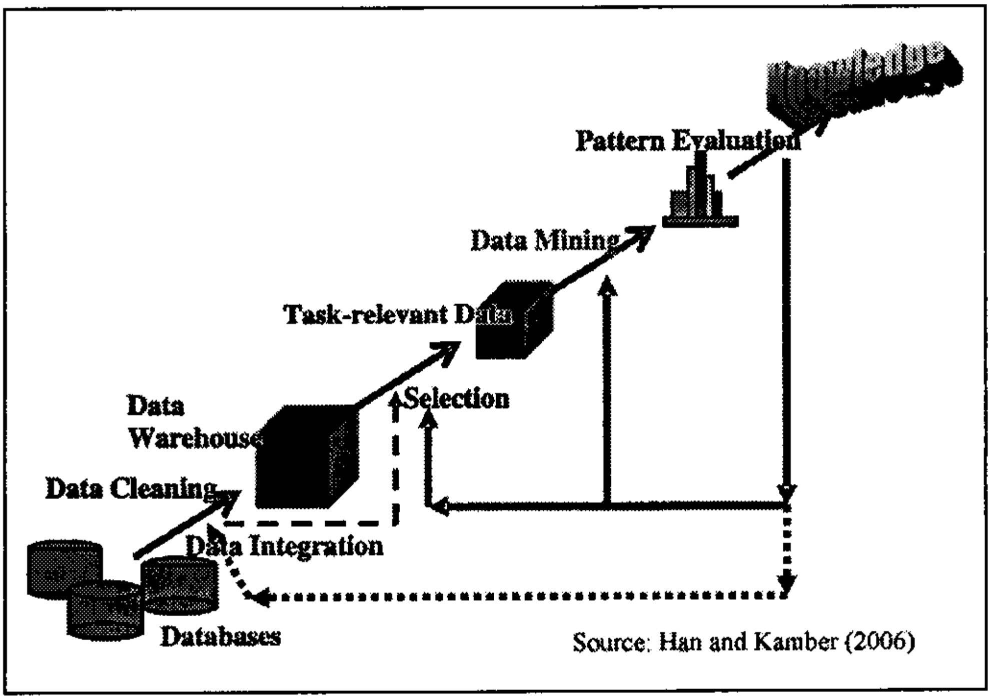

FIG. 1 STEPS IN KNOWLEDGE DISCOVERY PROCESS

Data mining is an aggregation of many disciplines which include machine learning, statistics, database technology, information science and visualization. Techniques like neural networks, support vector machines, fuzzy and rough set theory from other disciplines are often used depending upon the data mining approach used. Data mining can be performed on any kind of data repository, ranging from relational databases to World Wide Web. Generally, data mining models are classified into two broad categories: descriptive and predictive. While the former characterize the general properties of the data in the database thereby helping in understanding the underlying processes, the latter carry out inferences from the current data, using some equation or set of rules, thereby making it possible to make predictions regarding an unseen or unmeasured value (the dependent variable or output) from the known values (independent variables or input).

Temporal data mining deals with data mining of large sequential data sets. Sequential data implies data which is ordered with respect to some index. Time series is an example in which data is indexed by time. Considering the field of hydrology, there have been wide spread applications of temporal data mining models in the field of hydrology which ranges from the applications of artificial neural netwo.-ks, genetic algorithms, fuzzy set theory and rough set theory in the areas of rainfall-runoff modelling, streamflow forecasting and reservoir release optimization (Minns & Hall, 1996, Kumar et al., 2004, Kumar et al., 2007) to the applications of relevance vector machine and support vector machine in the areas of downscaling, prediction of rainfall (Tripathi et al., 2006, Anandhi et al., 2007, Ghosh et al., 2008). Also, the basic data mining functionalities like clustering and classification have been extensively applied for regionalization of watersheds, rainfall disaggregation and hydrologic prediction (Kumar et al., 2000, Rao et al., 2008). Regarding another basic data mining functionality called association rules and frequent patterns, not much work had been done in hydrology employing this concept, although it has been largely implemented in marketing, e-commerce, supply-chain management, group decision support systems and web usage mining. Frequent pattern mining or rule association has also been called market basket analysis because of its extensive application in the retail sales domain. There are many books that cover data mining techniques and various applications (Han & Kamber, 2006; Hand et al., 2001; Witten & Frank, 2000).

In this paper a brief summary of the various functionalities in data mining is presented in general, mentioning also the various algorithms used. The focus is given to a detailed description about the frequent patterns & association rules, its extraction, various algorithms used for rule extraction and also its applications in hydrology.

DATA MINING FUNCTIONALITIES

The various data mining functionalities and the variety of knowledge they discover are briefly presented below:

- Characterization: Data characterization is a summarization of general features of a target class of data. The data relevant to a user-specific class are normally retrieved by a database query. For example, one may want to characterize the Video Store customers who regularly rent more than 100 movies a year. The resulting description of the customer characteristics could become the general profile of the customers including their age, occupation, credit ratings etc. Simple methods of data summarization include measures of central tendency such as mean, median, mode and midrange and also measure of dispersion such as quartiles, inter-quartile range and variance. The outputs of the data characterization are usually presented in the forms of pies charts, bar plots, multidimensional tables, data cubes and also in the form of characteristic rules.

- Discrimination: Data discrimination is the comparison of the general features of objects between two classes referred to as the *target class* and the *contrasting class*. For example, the user may want to compare the general characteristics of the customers who rented more than 100 movies in the last year with those whose rental account is lower than 50. The methods used and also the form of output in data discrimination are very similar to that used for data characterization with the exception that data discrimination results include comparative measures.

- Frequent Patterns & Association analysis: Frequent patterns are the patterns that occur frequently in the data. For example, a set of items such as milk and bread that appear frequently together in a transactional data set is a frequent item-set. Finding such frequent patterns helps in mining associations and correlations in the data. Typical example of frequent pattern mining is market basket analysis. It helps in analyzing customer buying habits by finding associations between the different items the customer bought, thereby aiding retailers to develop specific marketing strategies. For example, the information that the customers who purchase computers also purchase printers along with it can be expressed as an association rule as below:

  Computer $\Rightarrow$ *Printer [Support = 5%, Confidence = 70%]*

  Rule support and confidence are the two measures of rule interestingness used in the above rule. A support of 5% means that in 5% of all the transactions in the data set, computer and printer are purchased together. A confidence of 70% implies that 70% of all the customers who bought computer also bought printer. Usually, of the numerous association rules discovered, the rules that are interesting are identified by setting a user defined minimum support threshold and a minimum confidence threshold.

- Classification: Classification analysis is the process of finding a model that organizes and distinguishes data into given classes, for the purpose of being able to use the model to predict the class of objects whose class label is unknown. Data classification is a two step process. In the first step which is called learning step (training phase), a classifier is built using a *training set* where all objects are already associated with known class labels. The classification algorithm learns from the training set and builds a model. The second step called test phase is for checking the accuracy of the classifier. Classifier is given a test set which is different from the training set. The accuracy is determined by the percentage of test set tuples that are correctly classified by the classifier. For example, for taking a loan issuing decision, the managers could analyze the customers' behaviours (e.g., age, properties and income), and label accordingly the customers who received loan with two possible labels "safe" and "risky". The classification analysis would generate a model that could be used to either accept or reject loan requests in the future, based on the customer characteristics. Various forms of classifiers are IF-THEN rules, decision trees, neural networks, Bayesian classifiers, support vector machines, genetic algorithms and fuzzy/rough set approaches.

- Prediction: While classification prefixes categorical labels, prediction is concerned with continuous valued functions. Prediction is more often referred to identify missing numerical values, or increase/ decrease trends in time related data. The overall idea is to use a large number of past values to estimate probable future values. Regression analysis is one common methodology used for numeric prediction.
- Clustering: Similar to classification, clustering is the organization of data into classes. However, in clustering, class labels are unknown and it is up to the clustering algorithm to discover acceptable classes. Clustering is also called *unsupervised classification*, because the classification is not dictated by given class labels. The objects are clustered based on the principles of maximizing the similarity between objects within the same class (*intra-class similarity*) and minimizing the similarity between objects of different classes (*inter-class similarity*). The clusters thus formed can be viewed as one group and thus clustering is a form of data compression. Major clustering methods include partitioning methods (k-means and k-medoids algorithms), hierarchical methods (agglomerative and divisive), density-based methods, grid based methods, model based methods and constraint based methods.
- Outlier analysis: Outliers are data elements that are completely different from or inconsistent with the remaining data set and thus cannot be grouped with the given class or cluster. Outliers can be the result of measurement error or even the inherent data variability. The outlier influence is minimized or eliminiated in many data mining algorithms. This could result in some loss of hidden information. Outlier detection and analysis called outlier mining can be done using various approaches like statistical distribution based outlier detection, distance based outlier detection, density based local outlier detection and deviation based outlier detection.
- Evolution and deviation analysis: Evolution and deviation analysis pertains to the study of time related data that changes with time. Evolution analysis models the evolutionary trends in data, which comply to characterizing, comparing, classifying or clustering of time related data. Deviation analysis, on the other hand, considers differences between measured values and expected values, and attempts to find the cause of the deviation from the anticipated values.

Temporal data mining tasks can be classified further as follows: (i) prediction (ii) classification (iii) clustering (iv) search & retrieval and (v) pattern discovery. Of the five, pattern discovery is of recent origin and is a typical temporal data mining task unlike the others which have been used in traditional data mining processes also. The idea of pattern discovery, terminology and various algorithms used are discussed here.

PATTERN DISCOVERY

The objective of pattern discovery is simply to unearth all patterns of interest. Unlike all other temporal data mining tasks which had their origins in other disciplines like machine learning and estimation theory, pattern discovery has its origin in data mining itself. In that sense, pattern discovery task is an exclusive data mining task. This section gives a brief introduction on the notions used in frequent patterns and also rule discovery.

Frequent Pattern: A pattern is a local structure in the data, typically a substructure or substring in the data. The purpose of pattern discovery task is to unearth all the 'interesting' patterns from the data. There are many ways to define a pattern in the data and also to assess the interestingness of the data, although there is no universal notion for it. Interestingness is usually measured in terms of the number of frequent patterns. A frequent pattern is one that occurs many times in the data. Many algorithms have been developed to formulate useful patterns and also to discover all the frequent patterns in the data.

Rule: Frequent patterns so discovered are used to discover useful rules. Rules have been popular representations of knowledge for many years. A rule consists of a pair of Boolean-values propositions namely a left-hand side proposition (or the antecedent) and a right-hand side proposition (or the consequent). The rule states that whenever the rule antecedent is true, then rule consequent will also be true. An example of rules in data mining is the association rule which is used to capture the correlations between different variables in the data (*Agrawal & Srikant,* 1994). Similarly, in the case of patterns, for a sequential data stream, if the pattern "B follows A" occurs n_1 times and the pattern "C follows B follows A" occurs n_2 times, then a temporal association rule "whenever B follows A, C will follow too" can be generated with a confidence of (n_2 / n_1). Interestingness of a rule is usually measured in terms of confidence and also the frequency of consequent i.e., high (n_2/n_1) and also high (n_2).

One of the early attempts for discovering patterns in sequential databases is by Wang et al. (1994) through a pattern discovery method. They employ this method to find frequent patterns for a large collection of protein sequences. Since, protein is a sequence of amino acids (commonly, 20 amino acids), for computational purpose proteins can be viewed as a symbolic sequences over an alphabet of size twenty. The method of Wang et al. (1994) finds some candidate segments by constructing a generalized suffix tree for a sample of the sequences from the full database. Then these are combined to construct candidate patterns and the database is searched for each of these candidate patterns using an edit distance based scoring scheme. The number of sequences in the database which are within the user-defined distance of a given candidate pattern is its final occurrence score and those patterns whose score exceeds the user-defined threshold are the resulting temporal patterns. However, this method may not be able to discover all the temporal patterns that meet the user-defined threshold constraints.

Agrawal & Srikanth (1994) proposed a basic generalized algorithm known as Apriori algorithm for association rule mining framework on a database of unordered transaction records. Since then there are numerous temporal pattern discovery algorithms that are developed following the Apriori logic. Hence it is appropriate here to first understand the working principle of Apriori before going into the other extensive works.

The Apriori Algorithm

Let the database contain customer transactions at an electronics store. In this a transaction is simply an unordered collection of items purchased by a customer in one visit. An item-set is a non-empty set of items. An item-set *i* can be denoted by $(i_1 i_2 .. i_j .. i_m)$, where i_j is an item. The above mentioned item-set *i* is an *m*-item-set since it contains *m* items. Each transaction in the database is an item-set. However, given an arbitrary item-set *i*, it may or may not be contained in the transaction *T*. Thus, the fraction of all transactions in the database in which the item-set *i* is contained is called the support of item-set *i*. If support of item-set *i* exceeds the user defined support threshold, then *i* is considered as a frequent item-set. These are the patterns of interest in this problem. Hence, the problem in Apriori algorithm is to discover all frequent item-sets in the database for a given user specified minimum support threshold.

Apriori employs a level wise iterative search, where *k* item-sets are used to explore *k+1* item-sets. First, the set of frequent 1-item-sets are found by scanning the database and accumulating the count for each item, and collecting those items whose count is more than minimum support threshold. The resulting set is denoted L_1. Next, L_1 is used to find L_2 the set of frequent 2-item-sets. This in turn is used to find L_3, and so on, until no more frequent *k*-item-sets can be found. However, finding of each L_k requires one full scan of database. In order to improve the efficiency an important property called Apriori property is used to reduce the search space.

Apriori property: All nonempty subsets of a frequent item-set must also be frequent. This can be explained as follows: Consider an item-set *i* which is not frequent i.e., number of occurrences of item-set *i*, $P(i) <$ *minimum support threshold*. Then, if an item *j* is added to the item-set *i*, then the resulting item-set i.e., $i \cup j$ cannot occur more frequently than *i*. Hence, is also not frequent. This property is introduced in the Apriori algorithm using a two step process. The steps below show how to find in the Apriori algorithm using a two step process. The steps below show how to find L_k from L_{k-1}.

Join step: A set of candidate *k*-item-sets, C_k is generated by joining L_{k-1} within itself. Let l_1 and l_2 be two item-sets in L_{k-1}. $l_i[j]$ denotes *j*th item in item-set l_i. The items within a transaction or item-set are arranged in lexicographic order. For the *(k-1)* item-set, this means that the items are sorted such that $l_i[1] < l_i[2] < \ldots < l_i[k-1]$. The join of members of L_{k-1} is possible if their first *(k-2)* items are common, i.e., item-sets l_1 and l_2 of L_{k-1} are joined if $(l_1[1] = l_2[1] \wedge l_1[2] = l_2[2] \wedge \ldots \wedge l_1[k-2] = l_2[k-2] \wedge l_1[k-1] < l_2[k-1]$. The condition $l_1[k-1] < l_2[k-1]$ ensures that no duplicates are generated. Thus, the resulting item-set formed by joining l_1 and l_2 is $l_1[1], l_1[2], \ldots, l_1[k-2], l_1[k-1], l_2[k-1]$.

Prune step: The members of C_k may or may not be frequent. To determine the count of each candidate in C_k, database is scanned and those candidates whose count is less than the minimum support threshold are pruned out. This would result in the determination of L_k. To reduce the size of C_k, the apriori property is used by removing any candidate *k*-item-set from C_k if any of its *(k-1)*-subset is not there in L_{k-1}.

There are two popular frameworks for frequent pattern discovery, namely sequential patterns and episodes.

Sequential Patterns

Agrawal & Srikanth (1995) extended the Apriori algorithm which was primarily meant for frequent item-sets to the case of temporal pattern in them. Unlike, the earlier database which consists of an unordered collection of transactions, each transaction in the database in this case carries a time stamp as well as a customer ID. The transactions made by a single customer will be a sequence of item-sets which is ordered by time. Finally the database would be having one such transaction sequence corresponding to each customer. Here is an example of sequence database with four customers. The set of items in the database is {a, b, c, d, e, f, g}. The transaction sequences of the four customers are as follows: (1) $\langle(\mathrm{a})(\mathrm{abc})(\mathrm{cf})\rangle$, (2) $\langle(\mathrm{ad})(\mathrm{b})(\mathrm{bc})\rangle$, (3) $\langle(\mathrm{ab})(\mathrm{df})(\mathrm{ef})(\mathrm{cd})\rangle$, (4) $\langle(\mathrm{g})(\mathrm{a})(\mathrm{c})(\mathrm{b})(\mathrm{c})\rangle$. Each customer transaction is enclosed in angular braces and each transaction is enclosed in round braces. Considering customer 2, he has made 3 transactions in which during first visit he bought items *a* and *d*, during second visit he bought item *b* and items *b* and *c* in third visit. There are 6 instances of items in sequence 1; so it has a length of six and hence it is a 6-sequence. In this item *a* occurs 2 times and contributes 2 to the length of the sequence. Yet, the entire sequence contributes only one to the support of $\langle \mathrm{a} \rangle$. Coming to the concepts of subsequences, the sequence $\langle(\mathrm{a})(\mathrm{bc})(\mathrm{c})\rangle$ is a subsequence of sequence 1 also the order of the events is preserved. Now consider a subsequence $s = \langle(ab)(c)\rangle$, which is a 3-pattern. We can see that only sequences 1 & 3 contain this subsequence. Therefore the support of *s* is equal to 2/4 = 0.5. It can be considered as a large pattern if the user specified minimum support threshold is less or equal to 0.5. Further, a sequence is considered as maximal, if it is not contained in any other sequence. In the sequence database given, all the sequences are maximal since each is not a subset of any other sequence. A sequence is considered as a sequential pattern if it is large and also maximal (among the set of all large sequences).

The mechanism of sequential pattern discovery is the same as of Apriori algorithm, starting with the discovery of all possible item-sets with sufficient support. Once all the large item-sets are found, the database is transformed by replacing each transaction by the large item-set. A set of new potentially large sequences called candidate sequences are generated. In the next step which is called the sequence phase, database is scanned for frequent patterns. The process of candidate generation and scanning goes on till no new frequent patterns are found.

Agarwal & Srikant (1994) proposed two families of algorithms called count-all and count-some algorithms, which are based on the Apriori algorithm. Count-all algorithm first counts all large sequences and then prunes out all non-maximal sequences. During the first pass all the large 1-sequences are obtained. 2-sequence candidates are constructed by combining the large 1-sequences in all possible ways. A second pass in the database identifies all large 2-sequences and the process goes on. In count-some algorithm, a maximality constraint is included to avoid all sequences which would contain longer sequences, since only maximal sequences are needed. This algorithm works in two phases – a forward phase in which all frequent sequences of certain lengths are found and a backward phase in which all the remaining frequent sequences are found out.

Many improvements have been done to these algorithms in order to improve their performance. The number of database passes required by the above algorithms is equal to the length of the longest sequential pattern. To improve the efficiency of these algorithms Zaki (1998) decomposed the original search space into smaller pieces using a lattice-theoretic approach. These smaller pieces can be independently processed thus reducing the number of passes needed. Shintani & Kitsuregawa (1998) have proposed parallel algorithms for sequential pattern discovery. Generalized Sequential Pattern (GSP) was developed by Srikant & Agrawal (1996) which uses a downward closure property of sequential patterns. They also incorporated some constraints such as user defined taxonomy of items as well as minimum and maximum time interval between items of a sequence. Sequential Pattern mIning with Regular expressIon consTraints (SPIRIT) proposed by Garofalakis et al. (2002) also economises the task by restricting to the user defined regular expressions. Algorithms economises the task by restricting to the user defined regular expressions. Algorithms specially developed for data having long frequent sequences were proposed by Pasquier et al. (1999) and Wang & Han (2004).

Frequent episodes

Another framework for discovering frequent patterns is using frequent episodes. Unlike the sequential pattern framework in which a collection of sequences are given and the task is to discover those sequences of items that occur sufficiently many times, here in the frequent episode framework, the data is a single long sequence and the task is to unearth all temporal patterns or episodes that occur frequently within the sequence. The concept of frequent episode discovery was first well-tried by Mannila et al. (1995), who applied this technique for analyzing alarm streams in a telecommunication network. In telecommunication network, there are different kinds of alarms that are triggered by different states of the network since the status of the network evolves dynamically with time. Mannila et al. (1997) used frequent episode discovery for alarm management system by ascertaining the relationships between different kinds of alarms, thus making it possible to foresee any network congestion and also helping to improve the network efficiency by giving early warnings. They have proposed a couple of algorithms for frequent episode discovery using various approaches. Before going into the details of the algorithms, the framework of frequent episode discovery is explained first.

Framework of frequent episode discovery

Event Sequence: The input data which is a sequence of events, referred to here as an *event sequence*, are denoted by $\langle (E_1, t_1), (E_2, t_2), \ldots\ldots, (E_i, t_i), \ldots\ldots \rangle$ where E_i takes values from a finite set of event types ε, and t_i is an integer denoting the time stamp of the ith event. The sequence is ordered with respect to the time stamps so that, $t_i \le t_{i+1}$ for all $i = 1, 2, \ldots$. A part of an event sequence is in Fig. 2 as an example with 6 event types A, B, C, D, E and F in it. The time period of the event sequence is from 9th day to 43rd day.

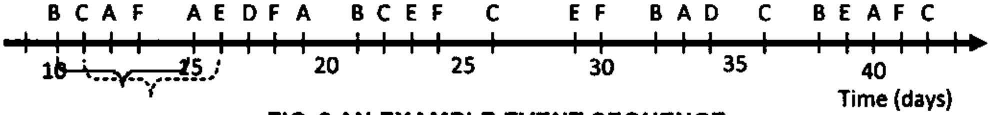

FIG. 2 AN EXAMPLE EVENT SEQUENCE

An event sequence s on can be expressed as a triple (s, T_s, T_e). Hence the above sequence S = (s, 9, 43), where $s = \langle (B,10),(C,11),(A,12),(F,13),(A,15),\ldots,(C,42) \rangle$, the time ordered sequence of events from beginning to end, T_s is the starting time, T_e is the ending time and $T_s \le t_i \le T_e$ for all $i = 1, 2, \ldots, n$.

Window: In this technique one is interested to find all frequent episodes from a class of episodes. For an episode to be interesting, the events in an episode must occur close to each other in time span. Usually, in almost all the algorithms, the user can define how close is close enough by defining a time window width within which the episodes should appear. A window can be defined as a slice of an event sequence and then the event is considered as a sequence of partially overlapping windows. A window on an event sequence (s, T_s, T_e) can also be expressed as a triple element $w = (w, t_s, t_e)$, where $t_s < T_e$, $t_e > T_s$ and w consists of those event pairs from s where $t_s \le t_i \le t_e$. The time span $t_e - t_s$ is called the width of the window w.

Consider the example of event sequence given above. Two windows of width 5 are shown. The first window starting at time 10 is shown in solid line, followed by a second window shown in dashed line. First window can be represented as $(\langle (B,10),(C,11),(A,12),(F,13)\rangle,10,15)$. Here the event $(A,15)$ occurred at the ending time is not included in the sequence. Similarly, the second window can be represented as $(\langle (C,11),(A,12),(F,13),(A,15)\rangle,11,16)$.

For a sequence S with a given window width "*win*", the total number of windows possible is given by $W(s,win) = T_e - T_s + win - 1$. This is because the first and last windows extend outside the sequence, such that the first window contains only the first time stamp of the sequence and the last window contains only the last time stamp.

Episode: Episode is a partially ordered collection of events occurring together. An episode α is defined by a triple element $(V_a, \varepsilon_a, g_a)$, where V_a is a collection of nodes, $\le_\alpha$ is a partial order on V_a and $g_a : V_a \rightarrow \varepsilon$ is a map that associates each node in the episode with an event type. Thus, an episode is a combination of events with a time-specified order. When there is a fixed order among the event types of an episode, it is called a *serial* episode and when there is no order at all, the episode is called a *parallel* episode.

An episode is said to *occur* in an event sequence if there exist events in the sequence occurring in exactly the same order as that prescribed in the episode, within a given time bound. For example, in the above sample event sequence, the events *(A, 19)*, *(B, 21)* and *(C, 22)* constitute an occurrence of a *3-node serial episode* $(A \rightarrow B \rightarrow C)$ while the events *(A, 12)*, *(B, 10)* and *(C, 11)* do not, because for this serial episode to occur, *A* must occur before *B* and *C*.

The frequency of an episode is defined as the number of windows in which the episode occurs divided by the total number of windows in the data set. An episode is considered frequent if its frequency is greater than or equal to the frequency threshold, min_fr specified by the user. The collection of all frequent episodes for the sequence s with the user defined win and min_fr is denoted by *F*(s, *win, min_fr*). Once the frequent episodes are discovered, rules can be extracted from this which describe the connection between the events in the event sequence. For example let a 2-node episode $\beta = A \rightarrow B$ occurs in 5% of the windows and the 3-node episode $\alpha = A \rightarrow B \rightarrow C$ occurs in 4% of the windows. Then a rule can be formulated as when a window with events A and B occurs, there is a chance of 0.8 that C will follow in the same window.

Mannila et al. has developed two algorithms WINEPI (Mannila et al., 1995) and MINEPI (Mannila & Toivonen, 1996) for finding out the frequent episodes.

WINEPI algorithm:

The methodology of Winepi consists of two key steps: (1) generation of potentially frequent episodes or candidate episodes and (2) actual counting of these candidates. The algorithm starts with an episode size, k=1. It has to make several passes through the database as the size k of the episodes increases. For k=1, each individual event belonging to the universal set of events, ε is considered as a potential pattern. Then k is increased by one unit at each pass. The algorithm terminates when there are no frequent patterns at the end of a pass or when there are no candidate episodes generated. The algorithm for winepi algorithm is given below.

```
Compute $C_1 = \{E \in \varepsilon\}$;
k:=1;
while $C_k \neq \Phi$ do
    // sequence set pass
    compute the pattern set $P_k = \{\alpha \in C_k \mid \text{freq}(\alpha) > \text{minFreq}\}$;
    k:= k+1;
    // candidate generation
    compute $C_k$ from $P_{k-1}$;
```

The initial candidate generation phase is based on a joining procedure which takes two patterns from the pattern set P_{k-1} to form a new episode α of size k. A pattern p_1 is joined with a pattern p_2 if the subepisode obtained by dropping the first event of p_1 is the same as the subepisode obtained by dropping the last event of p_2. At the end of this step, a set of potential candidates is generated. During the second step, the algorithm counts the occurrences of the candidate episodes of C_k in a single pass by using a sliding window approach. Winepi is based on a set $\mathcal{A}$ of state automata that accepts candidate episodes and ignores all other input. When a new event *E* enters in the window, an automaton *a* will be initialized for all the episodes beginning with *E* and will be added to $\mathcal{A}$. When the same event *E* leaves the window, *a* is removed from the automata.

Winepi had been applied to find patterns in telecommunication networks and rules found have been integrated in alarm handling software (Hatonen et al., 1996). It had also used to discover conserved patterns in protein sequences with some modification done on the algorithm (Ramstein et al., 2000). This algorithm allows users to express arbitrary unary constraints on individual events. Mannila & Toivonen (1996) expanded those constraints to include binary conditions on pairs of events.

MINEPI Algorithm

In this alternative approach called MINEPI, episode discovery is based on minimal occurrences of episodes. Instead of looking at the windows and only considering whether an episode occurs in a window or not, this approach looks at the exact occurrences of episodes and the relationships between those occurrences. One of the advantages of this approach is that focusing on the occurrences of episodes allows us to more easily find rules with two window widths, one for the left-hand side and one for the whole rule, such as "if *A* and *B* occur within 50 seconds, then *C* follows within 60 seconds".

For each frequent episode, the information about the locations of its minimal occurrences is stored. In the recognition phase, the locations of minimal occurrences of a candidate episode α are computed by a temporal join of the minimal occurrences of two sub-episodes of α. This is simple and efficient, and the confidences and frequencies of rules with a large number of different window widths can be obtained quickly, i.e., there is no need to rerun the analysis if one only wants to modify the window widths.

Minimal occurrences of episodes with their time intervals are identified in the following way. For a given episode α and an event sequence S, the minimal occurrence of α in S is the interval $[t_s, t_e]$, if (1) α occurs in the window $w = (w, t_s, t_e)$ on S, and if (2) α does not occur in any proper sub-window on w. A window $w' = (w', t'_s, t'_e)$ will be a proper sub-window of w if $t_s \leq t'_s$, $t'_e \leq t_e$, and width (w')< width(w). The set of minimal occurrences of an episode α in a given event sequence is denoted by $mo(\alpha) = \{[t_s, t_e) | [t_s, t_e)\}$.

As an example, consider the event sequence shown in Fig. 3.

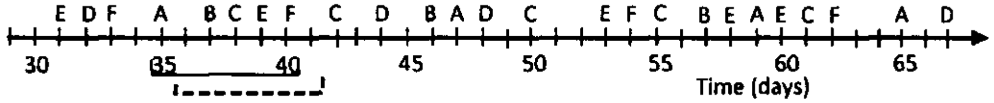

FIG. 3 AN EXAMPLE EVENT SEQUENCE SHOWING THE CONCEPT OF EPISODES

In this sequence the parallel episode α = A → B has four minimal occurrences i.e., mo(α) = {[35,38), [46,48), [47,58), [57,60)}. Note that, here the order of occurrence of events *A* and *B* doesn't matter since it is a parallel episode.

Further, a time bound episode rule is expressed as $\alpha[win_1] \Rightarrow \beta[win_2]$ i.e., if episode α has a minimal occurrence at interval $\{[\,t_s, t_e)$ with $t_e - t_s \leq win_1$, then episode β occurs at interval $\{[\,t_s, t'_e)$ with $t'_e - t_s \leq win_2$.

Mannila et al. (1997) had applied MINEPI algorithm on a variety of data sets such as alarm data, World Wide Web (WWW), texts and proteins.

A much closer work of these approaches was done by *Agrawal & Srikant* (1995) and Srikant & Agrawal (1996). These methods were also extended to find association rules by Han & Fu (1995) and Holsheimer et al. (1995). Laird (1993) gives a survey on patterns in sequential data. Winepi and Minepi approaches are further expanded by Harms et al. (2002) in their algorithms Representative Episodal Association Rules (REAR) and Minimal Occurences With Constraints And Time Lags (MOWCATL) respectively. These algorithms are able to incorporate constraints into temporal data.

MOWCATL algorithm

The Minepi approach was modified to handle separate antecedent and consequent constraints and maximum window widths and also the time lags between the antecedent and consequent to find natural delays embedded within the episodal relationships by Harms & Deogun (2004) in Minimal Occurrences With Constraints And Time Lags (MOWCATL) algorithm.

This algorithm first goes through the data sequence and stores the occurrences of all single events for the antecedent and consequent separately. The algorithm only looks for the target episodes specified by the user. So it prunes the episodes that do not meet the user specified minimum support threshold. Then two event episodes are generated by pairing up the single events so that the pairs of events occur within the prescribed window width and the occurrences of these two event episodes in the data sequence are recorded. This is repeated till there are no more events to be paired up. The process is repeated for three events, four events and so on until there are no episodes left to be combined that meet the minimum threshold. The frequent episodes for antecedent and consequent sequences are found independently. These frequent episodes are combined to form an episodal rule.

An episodal rule is that in which an antecedent episode occurs within a given window width, a consequent episode occurs within a given window width and the start of the consequent follows the start of the antecedent within a user specified time lag. For example, let episode X is of the events A and B, and episode Y is of the events C and D. Also the user specified antecedent window width is 3 months, consequent window width is 2 months and the time lag is 3 months. Then the rule generated would indicate that if A and B occur within 3 months, then within 3 months they will be followed by C and D occurring together within 2 months. The support of the rule is the number of times the rule occurs in the data sequence. The confidence of the rule is the conditional probability that the consequent occurs, given the antecedent occurs. For the rule "X is followed by Y", the confidence is the ratio of the Support[X and Y] and Support[X].

The support and confidence are the two measures used for measuring the interestingness of the rule. The values of these are set high to prune the association rules. The algorithm produces a good number of rules although the support and confidence are fixed high. The user needs some quantifying measures to select the most valuable rules. Several interestingness or goodness measures are used to compare and select better rules from the ones that are generated (Bayardo & Agrawal, 1999, Das et al., 1998, Harms et al., 2002). In MOWCATL algorithm, J-measure (Smyth & Goodman, 1991) is used for rule ranking. The J measure is given by

$$J(x;y)=p(x)\begin{bmatrix} p(y\mid x)\times\log[p(y\mid x)/p(y)]+ \\ [1-p(y\mid x)]\times\log\{[1-p(y\mid x)]/[1-p(y)]\} \end{bmatrix} \quad (1)$$

where p*(x), p(y)* and *p(y/x)* are the probabilities of occurrence of x, y and y given x respectively in the data sequence. The first term in the J-measure is a bias towards rules which occur more frequently. The second term inside brackets is well known as cross-entropy, namely the information gained in going from the prior probability *p(y)* to a posterior probability *p(y/x)* (Das et al., 1998). Compared to other measures which directly depend on the probabilities (Piatetsky-Shapiro, 1991), thereby assigning less weight to the rarer events, J-measure is better suited to rarer events since it uses a log scale (information based). As shown by Smyth and Goodman (1992), J-measure has the unique properties as a rule information measure and is a special case of Shannon's mutual information. The J values range from 0 to 1. The higher the J value the better it is.

MOWCATL algorithm was successfully applied to discover associations between climatic and oceanic parameters and drought indices in Nebraska by Tadesse et al (2005). The antecedent climatic indices used are Southern Oscillation Index (SOI), Multivariate ENSO Index (MEI), North Atlantic Oscillation (NAO), Pacific Decadal Oscillation (PDO) and Pacific/ North American Index (PNA) and the consequent drought indices used are Standardized Precipitation Index (SPI) and Palmer Drought Severity Index (PDSI).

APPLICATION OF TIME SERIES DATA MINING ALGORITHM TO INDIAN RAINFALL DATA

MOWCATL algorithm is applied to develop association rules of droughts and floods considering Indian monsoon rainfall as the consequent and relevant climatic indices as the antecedent.

Data used for the study

The time series data sets used in this study are of the monthly values for the period 1960 to 2005 and are defined as follows:

a. Summer monsoonal rainfall (June to September) for All India and also for the five homogenous regions (as defined by Indian Institute of Tropical Meteorology), for the period 1960 to 2005 (http://www.tropmet.res.in).
b. Darwin sea level pressure (DSLP) (NCEP, ftp.ncep.noaa.gov/pub/cpc/wd52dg/data/indices)
c. Nino 3.4, East central tropical pacific SST - 170° E – 120° W, 5° S – 5° N (http://www.cpc.ncep.noaa.gov/data/indices/sstoi.indices)
d. North Atlantic oscillation (NAO), normalized sea level pressure difference between Gibraltor and Southwest Iceland (http://www.cru.uea.ac.uk/cru/data/nao.htm).
e. 1×1 degree grid SST data over the region 40° E – 120° E, 25° S – 25° N (ICOADS, http://www.cdc.noaa.gov/icoads-las/servlets/datset)

Association rules for extremes

To find the spatial and temporal patterns of extreme episodes throughout the country, data mining algorithm is applied to rainfall data for All-India and its five homogenous regions (Northwest, Central Northeast, Northeast, West Central and Peninsular) rainfall data.

Selection of consequent episodes

In order to identify the extreme episodes, the rainfall for All India and for the five homogenous regions are divided into seven categories. The threshold values are determined by identifying the values at ±1.5, ±1 and ±0.5 standard deviations from the average. Threshold values calculated for each region are given in Table 1. The seven classes thus identified are named as: moderate drought, severe drought, extreme drought, normal rainfall, moderate flood, severe flood and extreme flood. For application of the algorithm, only the six extreme episodes (moderate drought, severe drought, extreme drought, moderate flood, severe flood and extreme flood) are specified as the target episodes.

Selection of antecedent episodes

1×1 degree grid SST data over the region 40° E – 120° E, 25° S – 25° N are averaged to a 5×5 degree grid data, thus reducing to 127 grids (excluding the land area regions). Among these, the most influencing grids are selected by plotting the correlation contour plots considering different lags for each region. The maximum correlation of SST with the summer monsoon is achieved at lag 7 for all the regions.

TABLE-1
THRESHOLD VALUES USED FOR THE CATEGORIZATION OF MONTHLY RAINFALL (MM) FOR VARIOUS REGIONS AND ALSO FOR ALL-INDIA

Region	Extreme Drought	Severe Drought	Moderate Drought	Normal rainfall	Moderate flood	Severe flood	Extreme flood
Northwest	≤150	150 X 500	500 X 850	850 X< 1550	1550 X 1900	1900 X 2250	2250
West-Central	≤1100	1100 X 1450	1450 X 1900	1900 X<2600	2600 X 3000	3000 X 3400	3400
Central Northeast	≤1200	1200 X 1600	1600 X 2000	2000 X<2850	2850 X 3300	3300 X 3700	3700
Northeast	≤2200	2200 X 2700	2700 X 3100	3100 X<3900	3900 X 4300	4300 X 4700	4700
Peninsular	≤1000	1000 X 1200	1200 X 1450	1450 X<1900	1900 X 2100	2100 X 2350	2350
All-India	≤1200	1200 X 1500	1500 X 1800	1800 X<2400	2400 X 2700	2700 X 2900	2900

* Rainfall in m

TABLE-2
THRESHOLD VALUES USED FOR THE CATEGORIZATION OF CLIMATIC INDICES (PREDICTORS)

Index	1	2	3	4	5	6	7
DSLP (mb)	≤ 6.0	6.0 X 7.5	7.5 X 8.5	8.5 X< 11.5	11.5 X 12.5	12.5 X 14.0	14.0
NAO	-2.5	-2.5<X -1.5	-1.5 X -1.0	-1.0 X<1.0	1.0 X 1.5	1.5 X 2.5	2.5
Nino3.4(°C)	≤25.5	25.5 X 26.0	26.0 X 26.5	26.5 X< 27.5	27.5 X 28.0	28.0 X 28.5	28.5

DSLP - Darwin Sea Level Pressure NAO - North Atlantic Oscillation Nino 3.4 - Nino 3.4 SST

The climatic indices which are used as antecedents in rule generation are DSLP, Nino 3.4, NAO and SST values of those grids which are showing maximum correlation with the summer monsoon rainfall of each region.

The climatic indices are also categorized into seven categories by segregating at ±1.5, ±1 and ±0.5 standard deviations from the average. Threshold values for these indices (except the SST grids) are given in Table 2. The time series algorithm employing the concepts of minimal occurrences with constraints and time lags was employed to find the associations between the antecedents and consequent. Climatic indices are considered as the antecedents and the target extreme episodes are considered as the consequents for generating the rules. A variety of window widths, time lags, frequency thresholds and confidence thresholds were tried to find the frequent episodes and rules. To assess the goodness (interestingness) of a rule, both confidence and J-measure were used.

RESULTS AND DISCUSSIONS

Association rules for drought

The data mining algorithm is applied to find the association rules for all the five regions and for All-India, based on the data from 1960 to 1982 (23 years). A confidence threshold of 0.7 and a minimum J-measure of 0.025 were used for the extraction of frequent rules. It is found that rules with maximum confidence level and J-measure were obtained for an antecedent window width of 4 months, consequent window width of 1 month and time lag of 7 months. The selected rules thus generated are shown in the Table 3.

For almost all regions, a combination of DSLP, NAO, Nino 3.4 and SST values of the respective most influencing grids is causing drought episodes of varying intensities. But the discrete states of the precursors are different for different regions. For example, a severe drought is occurring in West-Central region, if DSLP is between 12.5 and 14.0 (i.e., DSLP-6), NAO is between 1.0 and 1.5 (i.e., NAO-5), Nino 3.4 is between 28.0 and 28.5 (i.e., Nino-6) and lowest SST values (i.e., SST values which are less than 1.5 standard deviation from the average). Also, a combination of DSLP taking values between 12.5 and 14.0 (i.e., DSLP-6), NAO taking values between -1.5 and -1.0 (i.e., NAO-3), Nino 3.4 taking values between 27.5 and 28.0 (i.e., Nino-5) and lowest SST values are causing a moderate drought in the same region. For Peninsular region, if DSLP is between 12.5 and 14.0 (i.e., DSLP-6), NAO is between 1.0 and 1.5 (i.e., NAO-5), Nino 3.4 is between 28.0 and 28.5 (i.e., Nino-6) and lowest SST values at two grids (grids 21 and 26), then extreme drought is occurring with a confidence value of 1.0. Another most repeating rule in almost all regions is the combination of DSLP or NAO, Nino and SST as the precursors of drought. For All-India, a severe drought is preceded by a combination of NAO, Nino 3.4, and low

TABLE-3
SELECTED ASSOCIATION RULES FOR DROUGHT

Region	Rule no	Antecedent	Consequent	Confidence	J-Measure
Northwest	1	NAO-6, Nino-5, SSTgrid36-3, SSTgrid54-3	Severe drought	0.75	0.038
	2	DSLP-5, Nino-5, SSTgrid54-1, SSTgrid54-3	Moderate drought	0.75	0.0394
West-Central	1	DSLP-6, NAO-5, Nino-6, SSTgrid15-2	Severe drought	0.75	0.038
	2	DSLP-6, NAO-3, Nino-5, SSTgrid21-2	Moderate drought	1.0	0.043
Central Northeast	1	DSLP-6, Nino-6, SSTgrid21-2, SSTgrid56-2	Extreme drought	1.0 1.0	0.034 0.036
	2	DSLP-6, Nino-5, SSTgrid21-2, SSTgrid56-3	Severe drought	1.0	0.035
	3	NAO-3, NAO-6, Nino-6, SSTgrid56-3	Moderate drought		
Northeast	1	DSLP-5, NAO-3, NAO-5, SSTgrid105-2, SSTgrid123-2	Extreme drought	1.0	0.036
	2	DSLP-6, NAO-2, Nino-6, SSTgrid107-2, SSTgrid123-2	Severe drought	1.0	0.038
	3	NAO-7, SSTgrid105-3, SSTgrid107-3, SSTgrid123-2	Moderate drought	1.0	0.0484
Peninsular	1	DSLP-6, NAO-5, Nino-6, SSTgrid21-2, SSTgrid26-2	Extreme drought	1.0	0.039
	2	DSLP-6, Nino-5, Nino-6, SSTgrid26-3, SSTgrid31-2	Severe drought	1.0	0.038
	3	NAO-5, Nino-5, Nino-6, SSTgrid21-3, SSTgrid26-2	Moderate drought	0.75	0.038
All-India	1	NAO-5, Nino-5, Nino-6, SSTgrid72-3, SSTgrid74-3	Severe drought	0.75	0.031
	2	DSLP-5, Nino-5, SSTgrid73-2, SSTgrid74-1	Moderate drought	1.0	0.056

SST values and a moderate drought is preceded by a combination of DSLP, Nino 3.4 and low SST values. It can be noted that for all the repeating rules, the discrete states taken by the precursors are different for different regions.

The rules generated are clearly showing a negative relation with DSLP and Nino 3.4 and also a positive relation with the SST. But there is no such specific relation showing up with NAO. For example, rule(3) of Central Northeast region and rule(1) of Northeast region are showing both positive and negative NAO values as the precursors of drought episodes.

For some regions like Northwest, West-Central and All-India, no rules for extreme drought show up. The reason for this may be either no frequent episodes of antecedents are preceding the consequent or the rules for extreme drought are not above the given threshold values for confidence and J-measure.

Association rules for flood

The data mining algorithm is applied to find the association rules using the data from 1960 to 1982 specifying target episodes as moderately wet, severely wet and extremely wet. As in the previous case, rules with maximum confidence level and J-measure were obtained for an antecedent window width of 4 months, consequent window width of 1 month and time lag of 7 months. The selected rules thus generated are shown in the Table 4.

TABLE-4
SELECTED ASSOCIATION RULES FOR FLOOD

Region	Rule no	Antecedent	Consequent	Confidence	J-Measure
Northwest	1	DSLP-3, NAO-5, SSTgrid15-6, SSTgrid54-5, SSTgrid75-5	Extreme flood	1.0	0.038
	2	DSLP-1, NAO-6, Nino-2, Nino-3, SSTgrid75-5	Severe flood	1.0	0.0376
	3	DSLP-2, DSLP-3, NAO-3, NAO-6, SSTgrid36-5	Moderate flood	0.75	0.038
West-Central	1	NAO-3, NAO-6, SSTgrid21-5, SSTgrid27-6	Extreme flood	1.0	0.036
	2	DSLP-3, NAO-3, SSTgrid15-6, SSTgrid27-5, SSTgrid35-5	Severe flood	1.0	0.034
	3	DSLP-1, DSLP-2, NAO-1, SSTgrid35-5	Moderate flood	0.8	0.052

Region	Rule no	Antecedent	Consequent	Confidence	J-Measure
Central Northeast	1	DSLP-3, NAO-3, SSTgrid21-5, SSTgrid56-5, SSTgrid75-7	Extreme flood	1.0	0.035
	2	DSLP-3, NAO-3, Nino-3, SSTgrid21-5, SSTgrid75-6	Severe flood	1.0	0.042
	3	DSLP-1, Nino-2, Nino-3, SSTgrid35-5	Moderate flood	1.0	0.060
Northeast	1	DSLP-2, NAO-6, Nino-3, SSTgrid107-5, SSTgrid123-6, SSTgrid124-5	Severe flood	1.0	0.038
	2	DSLP-1, DSLP-2, NAO-2, SSTgrid124-5	Moderate flood	0.75	0.036
Peninsular	1	NAO-3, Nino-3, SSTgrid10-5, SSTgrid21-5, SSTgrid26-6	Severe flood	0.75	0.043
	2	DSLP-1, NAO-6, SSTgrid10-5, SSTgrid26-6	Moderate flood	1.0	0.029
All-India	1	DSLP-2, NAO-1, NAO-2, SSTgrid37-6	Extreme flood	1.0	0.058
	2	DSLP-2, NAO-6, SSTgrid37-5, SSTgrid72-5, SSTgrid73-5	Severe flood	0.83	0.063
	3	DSLP-2, NAO-7, Nino-3, SSTgrid37-5, SSTgrid72-5	Moderate flood	0.75	0.037

The combination of precursors is different for each region, with indices DSLP and NAO appearing in rules for almost all regions. A combination of DLSP, NAO and high SST values are causing flood of varying intensities in almost all the regions. Considering All-India rainfall, higher SST conditions, DSLP value between 6.0 and 7.5 (i.e., DSLP-2) and NAO value less than -1.5 (i.e., NAO-1 and NAO-2) occurring within 4 months is succeeded by extreme flood at a lag of 7 months. Severe flood is preceded by a combination of the higher SST conditions, a DSLP value between 6.0 and 7.5 (i.e., DSLP-2) and NAO value between 1.5 and 2.5 (i.e., NAO-6) occurring within 4 months. Rules generated for flood show a negative correlation of rainfall with DSLP and Nino 3.4. Here also, rules for extreme flood did not show up for Northeast and Peninsular regions.

Validation of the rules

In order to validate and to check the consistency of the rules generated, data mining algorithm is again used to generate rules for drought and flood using the data for the subsequent 23 years (1983 – 2005). The threshold values for confidence and J-measure are kept same as for rule extraction. As in training, rules with maximum confidence level and J-measure were obtained for an antecedent window width of 4 months, consequent window width of 1 month and time lag of 7 months.

A comparison of the rules generated during the calibration period and the validation period shows that almost all the rules for both drought and flood are following the same combination of antecedents for the corresponding consequent with slight change in the values of confidence and J-measure. The variations in these interestingness measures are mainly due to the difference in the number of consequent episodes occurring in the calibration and validation periods.

A considerable deviation from the calibration rules is only in the association rules for drought for Central Northeast and Peninsular region, in which different SST grids are showing low SST values in the validation period. Instead of grid-56 showing a low SST value in rule(2) of Central Northeast, grid-38 is showing a low SST value during validation period. Also, for rule(3), grid-56 is replaced by grid-75. Similarly, in rule(1) of Peninsular region, grid-21 is replaced by low SST values of grid-31. For all other regions, the combinations are exactly the same for both drought and flood rules, with only a slight deviation in the discrete classes taken by the antecedents.

A comparison of the values taken by these indices reveals that they are taking nearby discrete classes during calibration and validation periods. Similarly, analyzing drought rules for other regions and also flood rules, it can be seen that even if the rules are not satisfying exactly, nearby classes of the indices specified in the rules are always preceding the target episodes. This necessitates the need for a flexible allotment of the classes for the indices. Instead of defining the classes with abrupt and well defined boundaries, a vague and ambiguous boundary by making use of the concept of fuzzy sets, can be used for classifying the indices into different sets.

CONCLUSIONS

Data mining is a powerful technology to extract the hidden predictive information from databases thus helping in the prediction of future trends and behaviors. Among the data mining techniques, the temporal data mining field is of recent origin. In this article, algorithms related to temporal data mining are briefed. Apriori algorithm, from which almost all pattern discovery algorithms originated, is discussed in detail and the different studies conducted over time by various authors are also discussed.

The above algorithm is adopted by the authors to find the spatial and temporal patterns of extreme rainfall events throughout India by considering the rainfall for

All-India and for five homogenous regions of India. Such an implementation of extraction of association rules for the extreme conditions may help decision makers to improve their fundamental scientific understanding of droughts & floods, about their causes, predictability, impacts, mitigation actions, planning methodologies and policy alternatives.

Various rules generated for each region and for All-India clearly indicate a strong relationship with climatic indices chosen, i.e., DSLP, NAO, Nino 3.4 and SST values. From the rules extracted, it can be noticed that almost all the climatic indices mentioned above are occurring as antecedents for drought episodes, with different combinations and confidence values. However, for rules extracted for flood episodes, the combinations with Nino 3.4 are confronted only a few times.

REFERENCES

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases, In Proc. 20th Int. Conf. on Very Large Data Bases, pp.487– 499.

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns, In Proc. 11th Int. Conf. on Data Engineering, (Washington, DC), IEEE Comput. Soc.

Anandhi, A., Srinivas, V.V., Nanjundiah, R.S. and Kumar, D.N. (2007). Downscaling precipitation to river basin in India for IPCC SRES scenarios using support vector machine, International Journal of Climatology, Vol.28, No.3, pp.401-420.

Bayardo, R.J. and Agarwal, R. (1999). Mining the most interesting rules, Proceedings of Fifth ACM International Conference on Knowledge Discovery and Data Mining, San Diego, CA, Association for Computing Machinery, pp.145-154.

ISL (1994). *Clementine User Guide and Reference Manual: A Data mining toolkit,* Integral Solutions Limited.

Das, G., Lin, K.I., Mannila, H., Ranganathan, G. and Smyth, P. (1998). Rule discovery from time series, Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, New York, pp.16-22.

Frawley, W.J., Piatetsky-Shapiro, G. and Matheus, C. J. (1991). Knowledge discovery in databases: An overview, Knowledge Discovery in Databases, pp.1-30, Reprinted in AI Magazine, Fall 1992.

Garofalakis, M., Rastogi, R. and Shim, K. (2002). Mining sequential patterns with regular expression constraints, IEEE Trans. Knowledge Data Eng., Vol.14, pp.530–552.

Ghosh, S. and Mujumdar, P.P. (2008). Statistical Downscaling of GCM Simulations to Streamflow using Relevance Vector Machine, Advances in Water Resources, Vol.31, No.1, pp.132-146.

Han, J. and Fu, Y. (1995). Discovery of multiple-level association rules from large databases, In Proceedings of the 21st International Conference on Very Large Data Bases (VLDB '95). Zurich, Swizerland, pp. 420–431.

Han, J., and Kamber, M. (2006). *Data Mining: Concepts and Techniques,* ISBN 1-55860-489-8, 550, Morgan Kaufmann Publishers.

Hand, D., Mannila, H. and Smyth, P. (2001). *Principles of data mining,* (Cambridge, MA: MIT Press)

Harms, S.K. and Deogun, J.S. (2004). Sequential association rule mining with time lags, Journal of Intelligent Information Systems, Vol.22, No.1, pp.7-22.

Harms, S.K., Deogun, J. and Tadesse, T. (2002). Discovering sequential rules with constraints and time lags in multiple sequences, Proc. 2002 Int. symposium on Methodologies for Intelligent Systems, Lyon, France, ISMIS, pp.432-441.

Hatonen, K., Klemettinen, M., Mannila, H., Ronkainen, P. and Toivonen, H. (1996). Knowledge Discovery from telecommunication network alarm databases, In 12th International Conference on Data Engineering (ICDE '96), pp.115-122.

Holsheiler, M., and Siebes, A. (1994). Data mining: *The search for knowledge in databases,* Technical Report CS-R9406, Centrum voor Wiskunde en Informatica.

Holsheimer, M., Kersten, M., Mannila, H., and Toivonen, H. (1995). A perspective on databases and data mining, In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD '95). Montr'eal, Canada, pp.150–155.

Kumar, D.N., Lall, U. and Peterson, M.R. (2000). Multi-site Disaggregation of Monthly to Daily Streamflow, Water Resources Research, American Geophysical Union, Vol.36, No.7, pp.1823-1833. DOI: 10.1029/2000WR900049.

Kumar, D.N., Raju, K.S. and Sathish, T. (2004). River Flow Forecasting using Recurrent Neural Networks, Water Resources Management, Springer, Vol.18, No.2, pp.143-161. DOI: 10.1023/B:WARM.0000024727.94701.12.

Kumar, D.N., Reddy, M.J. and Maity, R. (2007). Regional Rainfall Forecasting using Large Scale Climate Teleconnections and Artificial Intelligence Techniques, Journal of Intelligent Systems, Freund & Pettman, UK, Vol.16, No.4, pp. 307-322.

Laird, P. (1993). Identifying and using patterns in sequential data, Algorithmic Learning Theory, Fourth International Workshop, pp.1-18. http://people.brunel.ac.uk/%7Ehssrjis/issue/Volume 16 Abstracts/j16-4-3.htm

Mannila, H. and H. Toivonen (1996). Discovery of frequent episodes using minimal occurences, In Second International Conference on Knowledge Discovery and

Data Mining (KDD '96), AAAI Press, pp.146-151.

Mannila, H., Toivonen, H. and Verkamo, A.I. (1995). Discovery of frequent episodes in event sequences, In First International Conference on Knowledge Discovery and Data Mining (KDD '95), AAAI Press, pp.210-215.

Mannila, H., Toivonen, H. and Verkamo, A.I. (1997). Discovery of frequent episodes in event sequences, Data Mining Knowledge Discovery, Vol.1, pp.259–289.

Minns, A.W. and Hall, M. J. (1996). Artificial neural networks as rainfall runoff Models, Hydrological Sciences, Vol.41, No.3, pp.399-417.

Pasquier, N., Bastide, Y., Taouil, R. and Lakhal, L. (1999). Discovering frequent closed item-sets for association rules, In Lecture notes in computer science; Proc. 7th Int. Conf. on Database Theory (ICDT99), Jerusalem, Israel, Vol.1540, pp.398–416.

Piatetsky-Shapiro, G. (1991). *Discovery, analysis, and presentation of strong rules, Knowledge Discovery in Databases*, AAAI/M1T Press, pp.229-248.

Ramstein, G., Bunelle, P. and Jacques, Y. (2000). Discovery of Ambiguous Patterns in Sequences Application to Bioinformatics, Principles of Data Mining and Knowledge Discovery, Proceedings of Fourth European Conference, PKDD 2000, Lyon, France (Sept. 2000), pp.581-586.

Rao, A.R. and Srinivas, V.V. (2008). Regionalization of Watersheds: An Approach Based on Cluster Analysis, Series: Water Science and Technology Library, Vol.58, 241 p., ISBN: 978-1-4020-6851-5.

Shintani, T. and Kitsuregawa, M. (1998). Mining algorithms for sequential patterns in parallel: Hash based approach. In Proc. 2nd Pacific-Asia Conf. on Knowledge Discovery and Data Mining, pp.283–294.

Smyth, P., and Goodman, R. M. (1991). *Rule induction using information theory, Knowledge Discovery in Databases*, Cambridge: MA: The MIT Press, pp.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, Vol.4, No.4, pp.301-316.

Srikanth, R. and Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements, In Proc. 5th Int. Conf. on Extending Database Technology (EDBT), Avignon, France

Tadesse, T., Wilhite, D. A., Hayes, M.J., Harms, S.K. and Goddard, S. (2005). Discovering associations between climatic and oceanic parameters to monitor drought in Nebraska using data-mining techniques, Journal of Climate, Vol.18, No.10, pp.1541-1550.

Tripathi, S., Srinivas, V.V. and Nanjundiah, R.S. (2006). Downscaling of precipitation for climate change scenarios: A support vector machine approach, Journal of Hydrology, Vol.330, No.(3-4), pp.621-640.

Wang, J, Chirn, G.W., Marr, T.G., Shapiro, B., Shasha, D. and Zhang, K. (1994). Combinatorial pattern discovery for scientific data: Some preliminary results, In Proc. 1994 ACM SIGMOD Int. Conf. on Management of Data, Minneapolis, Minnesota, pp. 115-125.

Wang, J. and Han, J. (2004). BIDE: Efficient mining of frequent closed sequences, In 20th Int. Conf. on Data Engineering, Boston, MA

Witten, I.H. and Frank, E. (2000). *Data mining: Practical machine learning tools and techniques with JAVA implementations*, Morgan Kaufmann ,San Fransisco, CA.

Zaki, M. J. (1998). Efficient enumeration of frequent sequences, In Proc. ACM 7th Int. Conf. Information and Knowledge Management (CIKM).