Available online at www.sciencedirect.com

ScienceDirect

Omega 37 (2009) 16–28

omega
The International Journal of Management Science

www.elsevier.com/locate/omega

Policies for knowledge refreshing in databases[☆]

Xiao Fang, Ram Rachamadugu[*]

Information, Operations and Technology Management Department, College of Business Administration, University of Toledo, Toledo, Ohio 43606, USA

Received 7 October 2005; accepted 10 July 2006
Available online 25 September 2006

Abstract

Knowledge discovery in databases (KDD) provides organizations necessary tools to sift through vast data stores to extract knowledge. This process supports and improves decision making in organizations. In this paper, we introduce and define the concept of knowledge refreshing, a critical step to ensure the quality and timeliness of knowledge discovered in a KDD process. This has been unfortunately overlooked by prior researchers. Specifically, we study knowledge refreshing from the perspective of when to refresh knowledge so that the total system cost over a time horizon is minimized. We propose a model for knowledge refreshing, and a dynamic programming methodology for developing optimal strategies. We demonstrate the effectiveness of the proposed methodology using data from a real world application. The proposed methodology provides decision makers guidance in running KDD effectively and efficiently.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Decision support systems; Dynamic programming; Decision making/process; Artificial intelligence

1. Introduction

With the wide application of information technology in organizations, especially the rapid growth of E-Business, organizations have been accumulating vast amount of data on customers, sourcing partners, competitors, and business processes. Knowledge discovery in databases (KDD) [1] provides organizations necessary tools to sift through vast data stores to extract knowledge, which supports and improves organizational decision making. KDD is defined as a nontrivial process of discovering useful knowledge from data [1]. The KDD process consists of such steps as data pre-processing (i.e., data selection, cleaning and transformation), data mining (i.e., extracting patterns such as decision trees from preprocessed data sources) and pattern post-processing (i.e., evaluating and interpreting patterns to generate knowledge, such as classification rules extracted from a decision tree, that can support decision making) [1]. This is shown schematically in Fig. 1.

Earlier research applied KDD in functional areas such as finance [2–4], marketing [5,6] and operations management [7]. For example, Baesens et al. [2] applied neural network to solve credit-risk evaluation problems; Cooper and Giuffrida [5] used classification techniques to enhance a traditional market-response model; while Chen and Wu [7] developed an order batching approach based on association rule mining. Most of the prior KDD research assumed that data are static, and focused

[☆] This paper was processed by Associate Editor T. Teo.
[*] Corresponding author. Tel.: +1 419 530 2929; fax: +1 419 530 2290.
E-mail addresses: xiao.fang@utoledo.edu (X. Fang), rracham@utoledo.edu (R. Rachamadugu).

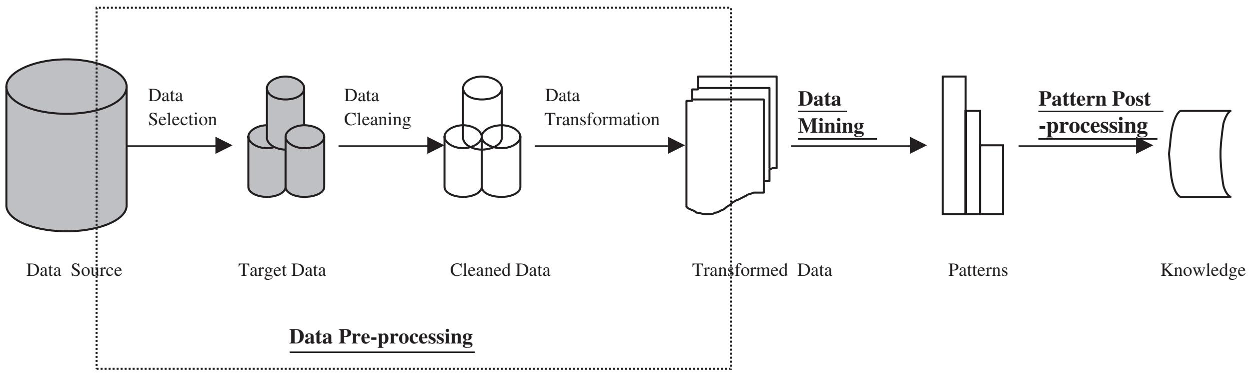


Fig. 1. The KDD process.

on either the efficiency of the KDD process (e.g., designing more efficient data mining algorithms) or identifying new applications of KDD. However, data are dynamic in reality (i.e., new data are continuously accumulated). Knowledge discovered using KDD becomes obsolete over time as the discovered knowledge only relates the nature of data at the time KDD process was run. Newly added data could bring in new knowledge and invalidate some earlier discovered knowledge. To support effective and efficient decision making, knowledge discovered using KDD needs to be updated along with its dynamic data source [5]. For example, Bourgeois and Eisenhardt [8] and Eisenhardt [9] pointed out that the use of obsolete information/knowledge in decision making is a major cause of poor and delayed strategic decisions. In this research, we focus on knowledge refreshing, which we define as the process of keeping knowledge discovered using KDD up-to-date with its dynamic data source.

2. Literature review

Prior research related to this work focused on maintaining patterns learned from data mining over a dynamic data source. Specifically, prior research related to this study can be grouped into two categories: incremental data mining and data stream mining.

Incremental data mining maintains patterns over a dynamic data source by revising patterns learned from a previous run of data mining, instead of learning from scratch. Several incremental data mining algorithms were proposed for major data mining models: classification, association rule mining and clustering. For classification, ID4 [10] and ID5 [11,12] were developed to revise a decision tree induced from old data as new data were added in. For association rule mining, the FUP algorithm [13] was first proposed to maintain large itemsets over a dynamic data source. A more efficient algorithm to maintain large itemsets over a dynamic data source was introduced by Thomas et al. [14]. Can [15] proposed an incremental clustering algorithm to maintain document clusters for an evolving document database.

Current data-intensive applications are based on very large data sources with rapid and continuous loading of large volumes of new data, termed as data streams. Examples of data streams include web logs, transaction databases in retail chains, network traffic data and financial tickers. Because of the processing speed and memory constraints, incremental data mining algorithms are not capable of maintaining patterns over data streams [16]. Recently, a number of data stream mining algorithms [17] were proposed to solve this emerging problem. Data stream mining algorithms usually employed data reduction techniques, such as sampling, to maintain approximate patterns over data streams. The Hoeffding tree algorithm, which applied the Hoeffding bound [18] to determine the number of samples required to learn a test attribute, was introduced in [16,19] to maintain approximate decision trees over data streams. The lossy counting algorithm [20] was proposed to maintain association rules over data streams. BIRTCH [21] is one of the early clustering algorithms suitable for data stream mining. The algorithm maintained a summary of a data source in memory, and then clustered the in-memory summary. Motivated by BIRTCH, a series of algorithms [22,23] were introduced for clustering data streams.

Both the incremental data mining research and the data stream mining research only addressed one step in the KDD process—the data mining step, and focused on maintaining patterns over a dynamic data source. However, it is knowledge, not patterns that can support organizational decision making effectively. For example, association rules (i.e., an example of patterns) learned using association rule mining are usually too many to be handled by decision makers [24]. Moreover, a substantial number of association rules are just common sense, or already known to decision makers. Hence these rules do not contribute to decision making [24]. To support effective decision making, the KDD process needs to be completed. For the above-mentioned example, the pattern post-processing step in the KDD process needs to be executed to extract knowledge (e.g., unexpected association rules) from all learned association rules, using various criteria and techniques [25,26].

The KDD process cannot be fully automated, except for the data mining step. It requires people (e.g., domain experts) in cleaning data before mining it and extracting knowledge from patterns discovered using data mining. Hence, the KDD is a costly process [20], as personnel costs dominate equipment and computational costs [27]. As a result, it may be impractical to run KDD whenever there is an update in a data source. Further, it may be unnecessary to run KDD whenever there is an update in a data source. Such a practice may often result in no new knowledge because successive snapshots of real world data sources are very likely to overlap considerably [28]. On the other hand, running KDD too seldom could result in losing critical knowledge. This will have adverse effect on decision making. Therefore, it is critical to determine when to run KDD so as to optimize the trade-off between the cost of knowledge loss and the cost of running KDD. Our research studies the knowledge refreshing problem from this perspective. Our paper is different from previous research on incremental data mining and data stream mining in the following ways:

(1) we consider the complete KDD process while past research focused on one step in the KDD process, i.e., the data mining step;
(2) this research studies when to refresh knowledge while past research investigated how to refresh patterns;
(3) earlier research investigated the knowledge refreshing problem from the standpoint of computer science, and focused on computational aspects of data mining algorithms. However, this research studies the knowledge refreshing problem from a managerial and business perspective—balancing the trade-off between the cost of poor decision making due to knowledge loss, and the cost of running KDD.

Other related works have studied optimal data acquisition (e.g., the data selection step shown in Fig.1) strategies for the KDD process. In these research works, two types of costs—the cost of data acquisition and the cost of misclassification errors, were considered when developing optimal data acquisition strategies. Some researchers [29–31] developed models and heuristics for data acquisition so as to minimize the cost of data acquisition, while others [32] proposed heuristics for data acquisition so that the cost of misclassification errors is minimized. Heuristics for optimizing the trade off between the cost of data acquisition and the cost of misclassification errors were also developed [33,34]. Our paper is different from previous research on optimal data acquisition strategies in the following ways:

(1) we consider the complete KDD process while past research focused on one step in the KDD process, i.e., the data acquisition step;
(2) this research proposes a model that can be applied to different data mining models while past research usually focused on one data mining model—classification;
(3) this research studies when to refresh knowledge while past research investigated optimal data acquisition strategies.

Rest of our paper is organized as follows. In the next section we propose a model for knowledge refreshing, and a dynamic programming methodology. Section 4 contains numerical illustration of the proposed optimal knowledge refreshing policy. We also discuss periodic knowledge refreshing policies, an alternative to the proposed policy. Finally, we provide conclusions and outline future research directions in the last section.

3. A model for knowledge refreshing

As shown in Fig. 2, a data source evolves over time as new data are continuously populated into it. By running KDD, knowledge is extracted from the data source and stored in a knowledge base. To keep the knowledge base up-to-date with the data source, a KDD run has to be executed, which incurs *the cost of running KDD*. There are queries submitted to the knowledge base to retrieve knowledge for various decision support purposes. For example, decision support systems retrieve knowledge from their knowledge bases to support organizational decision making, while recommender systems [35] installed in a large number of E-commerce websites such as Amazon.com retrieve knowledge to decide on product recommendations to consumers [36].

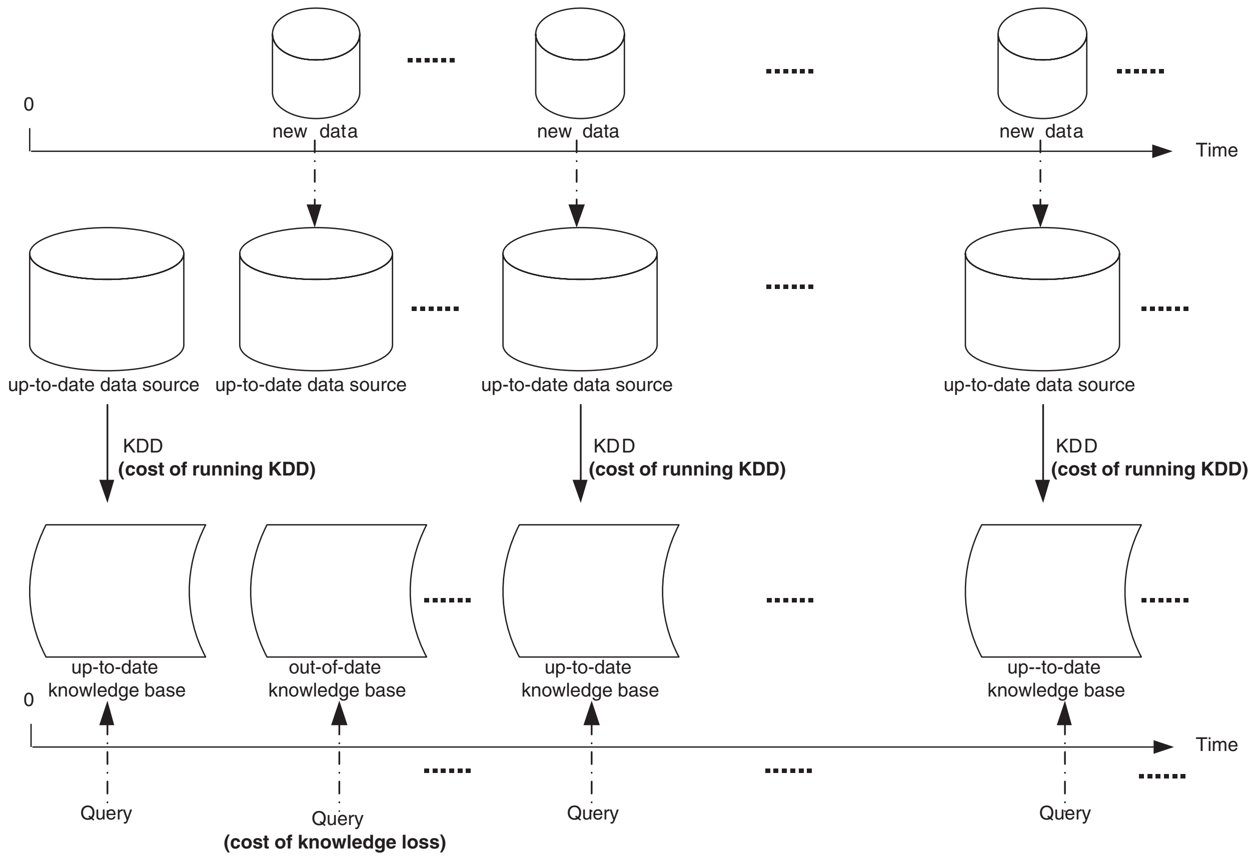


Fig. 2. Knowledge refreshing.

Table 1
Notation summary

Notation	Description
$h(d_i, d_j)$	The characteristics change between two data sets d_i and d_j, see formula (1)
h_e^m	The characteristics change at the *end* of time period m, see formula (2)
h_s^m	The characteristics change at the *start* of time period m
r_m	The knowledge-driven revenue generated during time period m
q_m	The number of queries submitted to a knowledge base during time period m
Z_m	The cost of knowledge loss over time period m, see formula (3)
S_n	The cost of running KDD at the start of time period n, see formula (4)
$C(i, j)$	The system cost from the start of time period i to the end of time period j, see formula (5)
$F(k)$	The optimal system cost from the start of the time horizon to the end of time period k, see formula (6)

Obviously, out-of-date knowledge also incurs cost, namely *the cost of knowledge loss*. For example, organizational decisions made based on out-of-date knowledge are ineffective or even wrong [8]; products recommended using out-of-date knowledge are not likely to be purchased since they are not attractive to consumers. The knowledge refreshing problem addressed in this research determines when, and how often to execute KDD so that the total cost, including costs of running KDD and costs of knowledge loss, over the time horizon is minimized. For the convenience of readers, notation used in this paper is summarized in Table 1.

3.1. The cost of knowledge loss

One of the key challenges in formulating our model for knowledge refreshing is to quantify the cost of knowledge loss. Knowledge loss is the phenomenon that partial or even all of the knowledge discovered in a previous run of KDD cannot be applied to current situations. The underlying reason for knowledge loss is the change in characteristics between the data used in the previous run of KDD and the data gathered in the current time period. Therefore, to quantify the cost of knowledge loss, we need to understand the deviation in characteristics between the data used in the previous run of KDD and the data in the current time period. Characterization of a data set is dependent on the data mining model used on the data set [28]. For example, a data set can be characterized using large itemsets mined from it if an association rule mining model was applied to the data set; a data set can be characterized using a decision tree mined from it if a decision tree induction model was applied to the data set; and a data set can be characterized using clusters discovered from it if a clustering model was applied to the data set.

Given two data sets d_i and d_j with characteristics c_i and c_j, respectively, characteristics change between d_i and d_j, $h(d_i, d_j)$ can be quantified as

$$h(d_i, d_j) = 1 - \frac{|c_i \cap c_j|}{|c_i \cup c_j|}. \tag{1}$$

Two types of characteristics deviations are captured in $h(d_i, d_j)$: characteristics appearing in d_i but absent in d_j, and characteristics absent in d_i but present in d_j. $h(d_i, d_j)$ has the following properties:

(1) $0 \leqslant h(d_i, d_j) \leqslant 1$; and the higher $h(d_i, d_j)$ the bigger the difference between the characteristics in d_i and those in d_j;
(2) $h(d_i, d_j) = 0$ if and only if $c_i = c_j$, which indicates that there is no characteristics change between d_i and d_j and all of the knowledge discovered in d_i can be perfectly applied to d_j;
(3) $h(d_i, d_j) = 1$ if and only if $c_i \cap c_j = \phi$, which indicates that the characteristics in d_i are totally different from those in d_j and none of the knowledge discovered in d_i can be applied to d_j;
(4) $0 < h(d_i, d_j) < 1$ if and only if $c_i \neq c_j$ and $c_i \cap c_j \neq \phi$, which indicates that there is some difference between the characteristics in d_i and those in d_j; hence only some of the knowledge discovered in d_i can be applied to d_j.

Note that we apply set operations (i.e., intersection and union) on c_i and c_j in (1). c_i and c_j, discovered using any of the three major data mining models [37], are actually sets or can be converted into sets [28]. Interested readers are directed to Ganti et al. [28] for further details. Following example illustrates how to estimate $h(d_i, d_j)$ for real world applications. Consider recommender systems used in many online retailers. Recommender systems are programs which predict items (e.g., books, music etc.) that a user might be interested in. According to [38], recommender systems in Amazon.com are powered by co-purchasing knowledge retrieved from a knowledge base constructed using previous purchasing transactions. Let d_1 be the data set of purchasing transactions from which the knowledge base is constructed, and d_2 be the data set of current purchasing transactions. Since the knowledge base consists of co-purchasing knowledge, large 2-itemsets can be used to characterize d_1 and d_2. Suppose that large 2-itemsets discovered from d_1 are $\{a, b\}$ and $\{b, c\}$, and large 2-itemsets discovered from d_2 are $\{b, c\}$ and $\{b, d\}$, where a, b, c, d represent products sold by Amazon.com. Hence, we have characteristics in d_1, $c_1 = \{\{a, b\}, \{b, c\}\}$, and characteristics in d_2, $c_2 = \{\{b, c\}, \{b, d\}\}$. According to (1),

$$h(d_1, d_2) = 1 - \frac{|c_1 \cap c_2|}{|c_1 \cup c_2|} = 2/3.$$

In this example, one co-purchasing pattern (i.e.,$\{a, b\}$) is absent in d_2 while another co-purchasing pattern (i.e., $\{b, d\}$) is present in d_2.

$h(d_i, d_j)$ can be used to quantify the cost of knowledge loss Z_m over a time period m. At the end of time period m, a sample data set d_m can be obtained. Let d_k be the data set used in the last KDD run before time period m. We denote h_e^m as characteristics change at the *end* of time period m and,

$$h_e^m = h(d_k, d_m). \tag{2}$$

h_e^m measures characteristics change between the data set used in the last KDD run and the data set around the end of time period m. Note that the value of h_e^m depends on when the last KDD run before time period m. We denote h_s^m as characteristics change at the *start* of time period m, which measures characteristics change between the data set used in the last KDD run and the data set around the start of time period m. h_s^m has the following properties:

(1) for two adjacent time periods, time period $m - 1$ and time period m, we have, $h_s^m = h_e^{m-1}$;
(2) $h_s^m = 0$, if the last KDD is run at the start of time period m or if there is no characteristics change at the end of time period $m - 1$, i.e., $h_e^{m-1} = 0$.

Let r_m be the knowledge-driven revenue generated during time period m. Knowledge-driven revenue is the revenue generated through the support of a knowledge base running in an enterprise. One example of knowledge-driven revenue is the amount of sales generated through a recommender system used by an online retailer. We assume that (1) queries are uniformly distributed throughout time period m, and that (2) characteristics changes are uniformly distributed over $[h_s^m, h_e^m]$ throughout time period m. We have the following Lemma.

Lemma 1. *The cost of knowledge loss Z_m over time period m is*,

$$Z_m = \begin{cases} r_m \dfrac{\ln(1-h_e^m+\varepsilon)-\ln(1-h_s^m+\varepsilon)}{h_s^m-h_e^m} - r_m & \textit{if } h_s^m \neq h_e^m, \\ \dfrac{r_m}{1-h_e^m+\varepsilon} - r_m & \textit{if } h_s^m = h_e^m, \end{cases} \qquad (3)$$

where factor ε is a very small nonnegative number less than 1.

Proof. Let q_m be the number of queries submitted to a knowledge base during time period m (see Fig. 2). Consider a small interval $(t, t+\Delta t)$ over time period m, where $0 \leqslant t \leqslant 1$. By assumption (1), the number of queries arriving during the interval $(t, t+\Delta t)$ is $q_m \Delta t$. Assuming that each query contributes the same to the knowledge-driven revenue generated during time period m, the knowledge-driven revenue generated during the interval $(t, t+\Delta t)$ is, $q_m \Delta t \frac{r_m}{q_m} = r_m \Delta t$.

(1) $h_s^m \neq h_e^m$: By assumption (2), characteristics change at time t is $h_s^m + (h_e^m - h_s^m)t$, which measures characteristics change between the data set used in the last KDD run and a sample data set obtained at time t. Note that the knowledge-driven revenue $r_m \Delta t$ is generated under the condition that there is characteristics change of $h_s^m + (h_e^m - h_s^m)t$. If there were no such characteristics change, the expected knowledge-driven revenue generated during the interval $(t, t+\Delta t)$ would be $r_m \Delta t / (1 - h_s^m - (h_e^m - h_s^m)t + \varepsilon)$, where factor ε is a very small nonnegative number less than 1. In other words, if the knowledge discovered in the last KDD run could be perfectly applied to current situations over the interval $(t, t+\Delta t)$ (i.e., there were no knowledge loss over the interval), the expected knowledge-driven revenue generated during the interval would be $r_m \Delta t / (1 - h_s^m - (h_e^m - h_s^m)t + \varepsilon)$. Factor ε is introduced to emulate the situation that a very small portion of the knowledge-driven revenue could be generated even when knowledge retrieved from a knowledge base are obsolete. For example, if a person plans to purchase pillow and comforter during her visit to Target.com, after putting pillow in her shopping cart, the person will purchase comforter irrespective of the type of recommendations from Target.com. The value of ε is application specific. Usually, the value of ε for companies selling commodities is higher than the value of ε for companies selling innovative products. Products sold by the former companies are already familiar to customers and their purchases depend less on the quality of knowledge-based recommendations; while products sold by the later companies are new to customers and their purchases depend more on the effectiveness of knowledge-based recommendations. Hence, the value of ε for firms selling commodities is higher than the value of ε for firms selling innovative products. Although determining the value of ε is beyond the scope of the paper, we show in Section 3 the effectiveness of our methodology under a series of different values of ε. The cost of knowledge loss over the interval $(t, t+\Delta t)$ is,

$$\frac{r_m \Delta t}{1 - h_s^m - (h_e^m - h_s^m)t + \varepsilon} - r_m \Delta t$$
$$= \frac{r_m \Delta t}{1 - h_s^m - (h_e^m - h_s^m)t + \varepsilon}[h_s^m + (h_e^m - h_s^m)t - \varepsilon].$$

As $\Delta t \to 0$, the cost of knowledge loss over time period m is,

$$\int_0^1 \frac{r_m[h_s^m + (h_e^m - h_s^m)t - \varepsilon]}{1 - h_s^m - (h_e^m - h_s^m)t + \varepsilon}\,\mathrm{d}t$$
$$= r_m \frac{\ln(1-h_e^m+\varepsilon)-\ln(1-h_s^m+\varepsilon)}{h_s^m-h_e^m} - r_m.$$

(2) $h_s^m = h_e^m$: Under this situation, by assumption (2), characteristics changes throughout time period m are constant with the value of h_e^m. Similar to (1), the knowledge-driven revenue $r_m \Delta t$ is generated under the condition that there is characteristics change of h_e^m. If there were no such characteristics change, the expected knowledge-driven revenue generated during the interval $(t, t+\Delta t)$ would be $r_m \Delta t / (1 - h_e^m + \varepsilon)$. The cost of knowledge loss over the interval $(t, t+\Delta t)$ is given by

$$\frac{r_m \Delta t}{1 - h_e^m + \varepsilon} - r_m \Delta t = \frac{r_m \Delta t}{1 - h_e^m + \varepsilon}(h_e^m - \varepsilon).$$

As $\Delta t \to 0$, the cost of knowledge loss over time period m is,

$$\int_0^1 \frac{r_m(h_e^m - \varepsilon)}{1 - h_e^m + \varepsilon} \mathrm{d}t = \frac{r_m}{1 - h_e^m + \varepsilon} - r_m.$$

The cost of knowledge loss Z_m defined in formula (3) can be explained as the difference between the potential knowledge-driven revenue generated under the condition that there is no knowledge loss and the actual knowledge-driven revenue r_m. According to (3), if $h_s^m \neq h_e^m$, the potential knowledge-driven revenue generated under the condition that there is no knowledge loss is $r_m(\ln(1 - h_e^m + \varepsilon) - \ln(1 - h_s^m + \varepsilon))/(h_s^m - h_e^m)$; and if $h_s^m = h_e^m$, the potential knowledge-driven revenue generated under the condition that there is no knowledge loss is $r_m/(1 - h_e^m + \varepsilon)$.

3.2. The model and solution procedure

We propose a finite-time model for the knowledge refreshing problem. As shown in Fig. 3, the time horizon of the model is labeled as $t = 0, 1, \ldots, n, \ldots, N$, with 0 being the starting time of the time horizon and n, where $1 \leqslant n \leqslant N$, denoting time period n (e.g., day n). We denote q_n as the number of queries submitted to a knowledge base during time period n, and r_n as the knowledge-driven revenue generated during time period n. At the beginning of each time period, an action needs to be decided on a binary action set $A = \{0, 1\}$, where 0 denotes not running KDD, and 1 denotes running KDD. Assuming that the knowledge base and the database are synchronized at the start of the time horizon, there is no need to run KDD at the start of the time horizon (or at the start of time period 1) and the first decision is made at the start of time period 2. The objective is to minimize the system cost over the time horizon. The system cost consists of two parts: costs of knowledge loss and costs of running KDD.

Let S_n be the cost of running KDD at the start of time period n, where $1 \leqslant n \leqslant N$. As discussed above, there is no need to run KDD at the start of time period 1 and we set S_1 to be 0. Although the computational cost of data mining is proportional to the size of the mined data, the cost of data pre-processing and pattern post-processing, which usually involves people, typically does not vary much with data volume. The latter dominates the cost of running KDD. Hence we model the cost of running KDD as a constant S. However, note that the model can easily be generalized to vary S_n for different time periods. For real world applications, S can be estimated using the number of persons involved in a KDD project times their pay rates.

$$S_n = \begin{cases} 0 & \text{if } n = 1, \\ S & \text{if } 1 < n \leqslant N. \end{cases} \tag{4}$$

An exhaustive enumeration of all possible decision combinations over the time horizon to search for the optimal one requires a time complexity of $O(2^N)$. The exponential growth in the computational burden of the solution discourages its application. However, we propose an efficient dynamic programming approach, which requires only a time complexity of $O(N^2)$ to find the optimal solution. Let $C(i, j)$ be the system cost from the start of time period i to the end of time period j, where $1 \leqslant i, j \leqslant N$ and $i \leqslant j$, given that between the start of time period i and the end of time period j KDD is only run at the start of time period i. We have,

$$C(i, j) = S_i + \sum_{m=i}^{j} Z_m. \tag{5}$$

In (5), S_i can be calculated using (4) and Z_m can be calculated using (3).

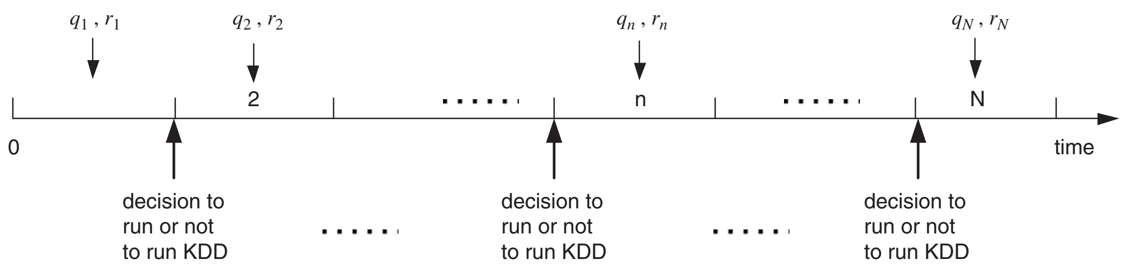
Let $F(k)$ be the optimal system cost from the start of the time horizon to the end of time period k, where $k = 1, 2, \ldots, N$. Suppose that the last KDD run before the end of time period k occurred at the start of time period l, where $1 \leqslant l \leqslant k$. By the principle of dynamic optimality [39],

$$F(k) = \min_{l=1,\ldots,k} \{F(l - 1) + C(l, k)\}. \tag{6}$$

In (6), initial value $F(0)$ is 0. Applying (6), $F(N)$ can be solved forward from the start of the time horizon to time period N. At the same time, the optimal decision sequence for running KDD over the time horizon is obtained. We will illustrate our methodology with an example in the next section.

4. Practical illustration

The purpose of this numerical illustration is to demonstrate the usefulness and the effectiveness of the proposed methodology using data from a real world application. We employ a database collected by the Internet marketing company ComScore Media Matrix (www.comscore.com), which records consumer online searching and purchasing behavior. The company uses a client-side program installed on the recruited households' home computers to collect detailed website



q_1, r_1
q_2, r_2
q_n, r_n
q_N, r_N
2
n
N
0
time
decision to run or not to run KDD
decision to run or not to run KDD
decision to run or not to run KDD

Fig. 3. A finite-time model for knowledge refreshing.

visiting data, transaction data (i.e., records of online purchasing) and demographic data of these households. We focus in particular on transaction data with an online retailer in this illustration. We chose the online retailer because a large number of transactions are generated every day at its website. Also, online retailers employ intensive KDD applications [36]. The data set used in this research captured 14,547 transactions with the online retailer, conducted by 6355 households across the United States during a 6-month time period from July 2002 to December 2002. Each transaction consists of the following fields: ID of Household, Date of Transaction, Product Name, Product Price and Basket Total (i.e., total dollar amount of a transaction).

The application considered in this research is a recommender system used in the online retailer. The recommender system is powered by co-purchasing knowledge retrieved from a knowledge base constructed using previous purchasing transactions. Assume that the knowledge refreshing decision is made on a bi-weekly basis for this recommender system, and the cost of running KDD is $200. Note that the purpose of this section is to demonstrate the usefulness and the effectiveness of the proposed methodology, not necessarily to emulate the reality exactly. For example, the knowledge refreshing decision for the recommender system may be made on a daily basis, and the cost of running KDD, which can be estimated using the number of persons involved in a KDD project times their pay rates, could be different from $200 used in this illustration. We divide the 6-month transaction data into twelve equal time periods, labeled from time period 1 to time period 12, where each time period consists of approximately two weeks of transactions. Suppose that 5% of sales of the online retailer are contributed by the recommender system, the knowledge-driven revenue generated during a time period is 5% of the sum of basket total for all transactions in the time period. Table 2 lists knowledge-driven revenue generated during each of the 12 time periods.

Table 2
Knowledge-driven revenue in each time period (5% of sales)

Time period	Knowledge-Driven revenue ($)
1	2064.30
2	2189.95
3	2702.23
4	3357.95
5	2572.79
6	2544.82
7	2671.49
8	3260.48
9	3994.83
10	5603.92
11	8213.80
12	3683.61

For each of the 12 time periods, a sample data set can be obtained at the end of the time period and its h_e^m, characteristics change at the end of the time period, can be calculated using (2) and (1). Table 3 lists h_e^m for each of the 12 time periods. Next, the cost of knowledge loss over each time period can be calculated using (3). For example, let us calculate the cost of knowledge loss over time period 4 given that the last KDD run occurred at the start of time period 3 and factor ε is 0.0005. According to Table 2, the knowledge-driven revenue generated over time period 4 is $3357.95. According to Table 3, characteristics change at the end of time period 4 given that the last KDD run occurred at the start of time period 3, h_e^4, is 0.1 and characteristics change at the start of time period 4, h_s^4, which equals to characteristics change at the end of time period 3 given that the last KDD run occurred at the start of time period 3, is 0.04. Applying (3), the cost of knowledge loss over time period 4, given that the last KDD run occurred at the start of time period 3, is $252.06.

Table 3
Characteristics change at the end of each time period

Last KDD run at the start of time period k	h_e^m: characteristics change at the end of time period m											
	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$	$m=8$	$m=9$	$m=10$	$m=11$	$m=12$
Knowledge base and database are synchronized at the start of the time horizon	0.10	0.10	0.07	0.11	0.13	0.13	0.10	0.12	0.13	0.15	0.16	0.14
$k=2$		0.09	0.10	0.10	0.09	0.09	0.09	0.11	0.13	0.13	0.14	0.13
$k=3$			0.04	0.10	0.13	0.13	0.09	0.13	0.15	0.16	0.17	0.15
$k=4$				0.11	0.13	0.10	0.10	0.12	0.13	0.15	0.16	0.14
$k=5$					0.06	0.06	0.13	0.14	0.13	0.15	0.16	0.16
$k=6$						0.05	0.12	0.13	0.13	0.14	0.15	0.15
$k=7$							0.12	0.11	0.11	0.13	0.14	0.13
$k=8$								0.08	0.11	0.13	0.14	0.12
$k=9$									0.04	0.10	0.10	0.06
$k=10$										0.07	0.09	0.08
$k=11$											0.07	0.10
$k=12$												0.07

Now we can calculate system cost $C(i, j)$ using (5), where $1 \leqslant i, j \leqslant 12$. Table Table 4 list $C(i, j)$.

Applying (6), we have,

$$F(0) = 0,$$

$$F(1) = \min_{l=1}\{F(l-1)+C(l,1)\}=336.89 \text{ and } l=1,$$

$$F(2) = \min_{l=1,2}\{F(l-1)+C(l,2)\}=639.36 \text{ and } l=1,$$

$$F(3) = \min_{l=1,\dots,3}\{F(l-1)+C(l,3)\}=907.01 \text{ and } l=3,$$

$$F(4) = \min_{l=1,\dots,4}\{F(l-1)+C(l,4)\}=1222.09 \text{ and } l=3,$$

$$F(5) = \min_{l=1,\dots,5}\{F(l-1)+C(l,5)\}=1515.92 \text{ and } l=5,$$

$$F(6) = \min_{l=1,\dots,6}\{F(l-1)+C(l,6)\}=1706.91 \text{ and } l=5,$$

$$F(7) = \min_{l=1,\dots,7}\{F(l-1)+C(l,7)\}=2058.57 \text{ and } l=5,$$

$$F(8) = \min_{l=1,\dots,8}\{F(l-1)+C(l,8)\}=2436.25 \text{ and } l=8,$$

$$F(9) = \min_{l=1,\dots,9}\{F(l-1)+C(l,9)\}=2736.27 \text{ and } l=9,$$

$$F(10) = \min_{l=1,\dots,10}\{F(l-1)+C(l,10)\} = 3176.91 \text{ and } l=10,$$

$$F(11) = \min_{l=1,\dots,11}\{F(l-1)+C(l,11)\} = 3771.56 \text{ and } l=11,$$

$$F(12) = \min_{l=1,\dots,12}\{F(l-1)+C(l,12)\} = 4143.26 \text{ and } l=12.$$

Therefore, the optimal system cost is \$4143.26 and the optimal policy is to run KDD at the start of time periods 3, 5, 8, 9, 10, 11 and 12.

To demonstrate the effectiveness of the optimal knowledge refreshing policy derived using the proposed methodology, we compare the performance of the optimal policy and the performance of periodic knowledge refreshing policies, which have been popularly applied in practice. In a periodic knowledge refreshing policy, KDD is run at fixed intervals. For example, the periodic-2 knowledge refreshing policy runs KDD every 2 time periods. Fig. 4 shows performance comparison between the optimal knowledge refreshing policy and all possible periodic knowledge refreshing policies for the online retailer example, when factor ε, the cost of running KDD and percentage sales attributed to the use of the knowledge base are set at 0.0005, \$200 and 5%, respectively. The optimal knowledge refreshing policy reduces system cost by 7.1% if compared with the best periodic knowledge refreshing policy and it reduces system cost by 46.5% if compared with the worst periodic knowledge refreshing policy. On average, the optimal knowledge refreshing policy reduces system cost by 28.6%. The optimal knowledge refreshing policy consistently outperforms periodic knowledge refreshing policies for different values of factor ε. When factor ε is 0.001, the optimal knowledge refreshing policy reduces system cost by an average of 28.8%; when factor ε is 0.005 the optimal knowledge refreshing policy reduces system cost by an average of 29.6%. Detailed results for various values of ε are omitted for the sake of brevity.

Table 4
$C(i, j)$

	$C(i,j)$											
	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$	$j=9$	$j=10$	$j=11$	$j=12$
$i=1$	336.89	639.36	944.81	1360.85	1813.33	2300.63	2740.00	3232.72	3942.99	5094.68	6959.03	7767.06
$i=2$		322.99	666.79	1130.59	1464.06	1772.46	2096.20	2538.12	3192.59	4211.66	5818.77	6549.77
$i=3$			267.65	582.76	988.59	1440.94	1832.73	2332.56	3142.19	4406.06	6403.11	7275.71
$i=4$				457.47	909.96	1328.49	1697.47	2190.19	2900.47	4052.15	5916.50	6724.53
$i=5$					293.83	484.82	836.48	1474.75	2256.78	3408.46	5272.81	6126.79
$i=6$						280.61	590.61	1166.44	1888.24	2967.10	4735.61	5542.96
$i=7$							416.09	937.19	1533.52	2469.61	4076.73	4807.73
$i=8$								377.68	899.12	1835.21	3442.33	4125.52
$i=9$									300.02	806.21	1910.65	2305.95
$i=10$										440.65	1295.62	1715.16
$i=11$											594.65	1042.39
$i=12$												371.70

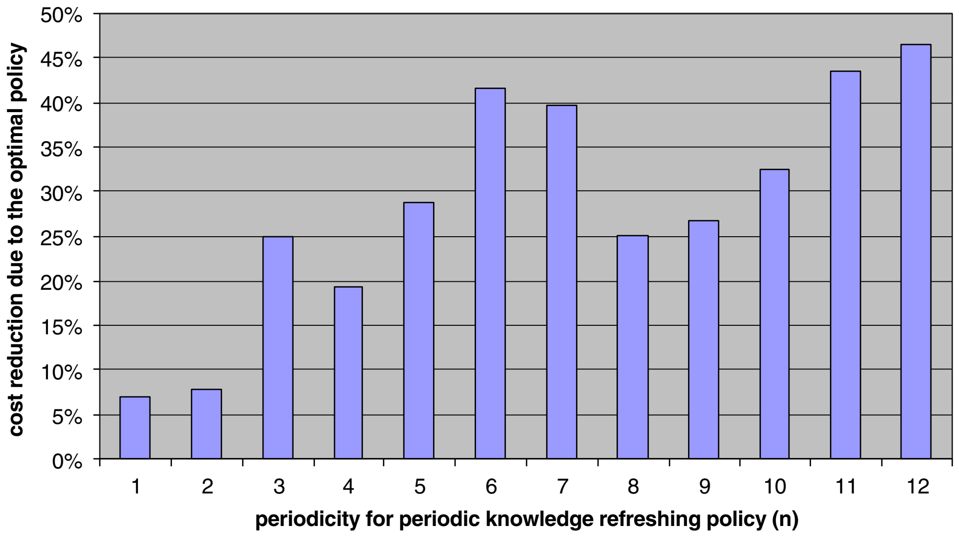


Fig. 4. Percentage cost reduction due to the optimal knowledge refreshing policy over periodic knowledge refreshing policies ($\varepsilon = 0.0005$, the cost of running KDD = \$200, and percentage sales attributed to the use of the knowledge base = 5%).

Setting factor ε and percentage sales attributed to the use of the knowledge base at 0.0005 and 5%, respectively, and increasing the cost of running KDD from \$0 to \$550, we compared the optimal knowledge refreshing policy and the best performing periodic knowledge refreshing policy. Fig. 5 shows the results. It is obvious that our optimal method provides the greatest benefits when the cost of running KDD, S, is neither too high nor too low, as is likely to be the case in practice. At extreme values of S, both the optimal policy and the best performing periodic policy would suggest very frequent or very infrequent running of KDD.

Fig. 6 graphically shows the performance of the optimal policy vis-à-vis the best performing periodic policy as percentage sales attributed to the use of knowledge base is varied. Here again, we notice that the benefits of using the optimal policy are low at extreme values. When percentage sales attributed to the use of knowledge base is very low or non-existent, neither policy would recommend refreshing knowledge base. Similarly, when percentage sales attributed to the use of knowledge base is high, both policies tend to refresh knowledge base at every opportunity to do so. It is in the intermediate range that the benefits of the

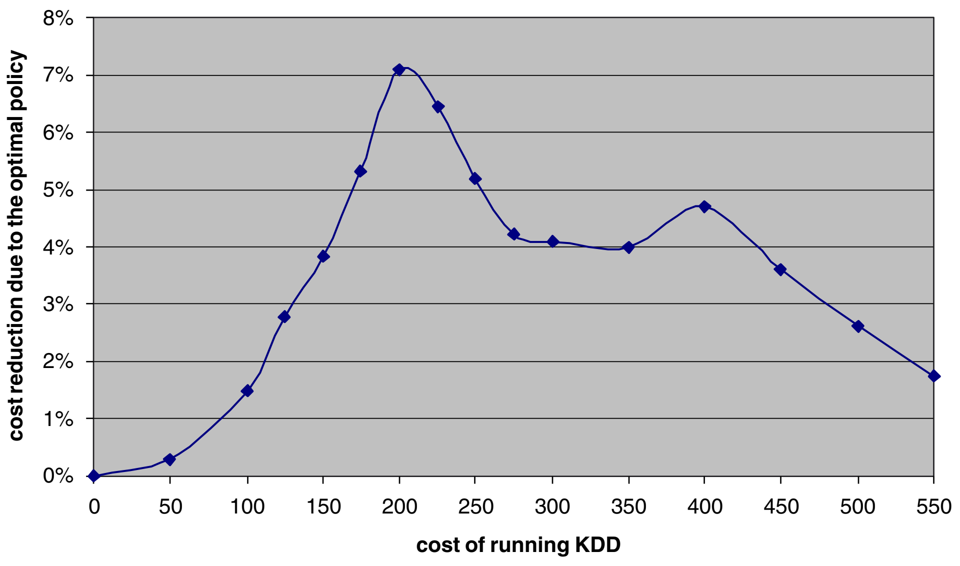


Fig. 5. Performance comparison between the optimal knowledge refreshing policy and the best performing periodic knowledge refreshing policies ($\varepsilon = 0.0005$, and percentage sales attributed to the use of the knowledge base $= 5\%$).

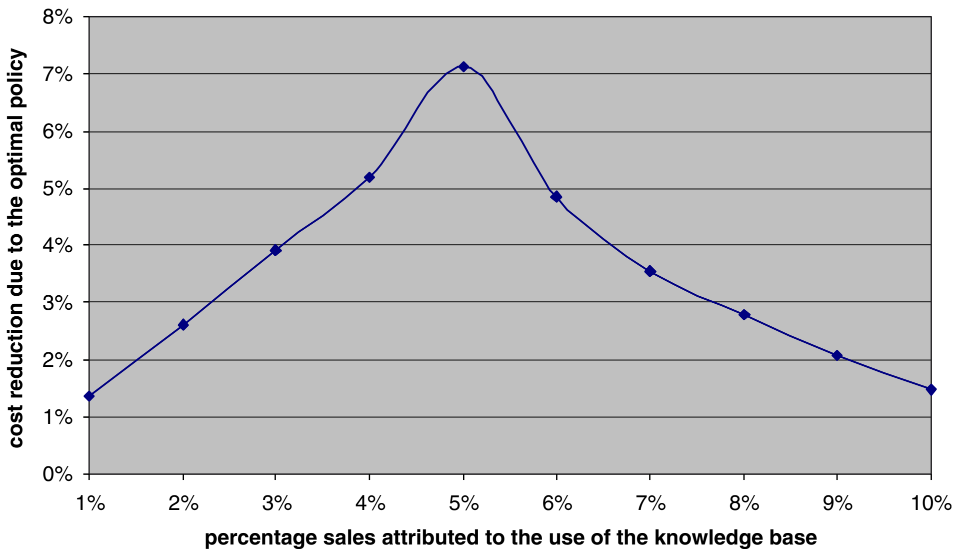


Fig. 6. Performance comparison between the optimal knowledge refreshing policy and the best performing periodic knowledge refreshing policies ($\varepsilon = 0.0005$, and the cost of running KDD $= \$200$).

optimal policy well exceed the best performing periodic policy.

5. Conclusions

In this paper we introduced and defined the concept of knowledge refreshing, a critical step to ensure the quality and timeliness of knowledge discovered in a KDD process. Unfortunately this had been overlooked by prior researchers. Specifically, we studied knowledge refreshing from the perspective of when to refresh knowledge so that the total system cost over a time horizon was minimized. We proposed a model for knowledge refreshing, and a dynamic programming methodology for finding optimal policies. The proposed dynamic programming methodology has low computational burden, and also low state space requirement. It can easily be implemented, in practice, based on parameters such as knowledge-driven revenue. We also illustrated the ease of implementation, and the effectiveness of the proposed methodology using data from a real world application.

Our research can be extended on several frontiers. The knowledge refreshing policy dictated by the dynamic programming approach requires forecast of model parameters such as change in characteristics of future transaction data, anticipated knowledge-driven revenue, and the cost of knowledge refreshing. Future research can be directed towards exploring the effect of forecast deviations on the performance of our methodology.

Since the number of queries, and the amount of new data arrivals in a period are only managerial estimates, another approach to the KDD problem would be to characterize these inputs as stochastic distributions. For example, the number of queries and/or data arrivals in a given period can be characterized by a probabilistic distribution such as Poisson or Normal distribution. Further, the cost associated with knowledge refreshing may not necessarily be known accurately in practice. It would be interesting to explore development of optimal policies for such situations.

We used characteristics change between data sets as a surrogate for knowledge change. However, it is conceivable that alternate quantifiers could be developed to measure the knowledge change. Since direct measurement of knowledge change is too costly, this avenue of research is merit worthy of future exploration in this area.

We used discrete time framework for developing optimal policies for knowledge refreshing in databases (similar to discrete time, dynamic lot sizing methods extensively used inventory theory). Our modeling was driven by the practical consideration that knowledge refreshing requires skilled professionals who may prefer predefined epochs for opportunities to run KDD processes. However, yet another approach for addressing the problem is to use the continuous time framework. For example, one can develop heuristic policies such as refreshing knowledge after certain number of queries, and/or certain number of new data updates occurs. For evaluating such policies, we need to develop optimal policies for continuous time framework. This additional avenue of research is merit worthy of future studies.

Finally, yet another area merit worthy of exploration is to find performance guaranties (average performance and/or worst case performance) for the best performing periodic knowledge refreshing policy. Performance guaranties for heuristics is a well explored area in Operations Research. Providing such guaranties can assure managers about the quality of policies used by them.

Acknowledgements

We thank the reviewers and the editors for their constructive comments and helpful suggestions on earlier versions of our paper. We also thank the Global Knowledge Management Center at the University of Utah for the access to ComScore Media Matrix data which was used in the practical illustration of the methodology developed in this paper.

References

[1] Fayyad U, Piatetsky-Shapiro G, Smyth P. The KDD process for extracting useful knowledge from volumes of data. Communications of the ACM 1996;39(11):27–34.

[2] Baesebs B, Setiono R, Mues C, Vanthienen J. Using neural network rule extraction and decision tables for credit-risk evaluation. Management Science 2003;49(3):312–29.

[3] Sarkar S, Sriram RS. Bayesian models for early warning of bank failures. Management Science 2001;47(11):1457–75.

[4] Tam KY. Neural network models and the prediction of bank bankruptcy. Omega 1991;19:429–45.

[5] Cooper LG, Giuffrida G. Turning data mining into a management science tool: new algorithms and empirical results. Management Science 2000;46(2):249–64.

[6] Gruca TS, Klemz BR. Using neural network to identify competitive market structures from aggregate market response data. Omega 1998;26:49–62.

[7] Chen MC, Wu HP. An association-based clustering approach to order batching considering customer demand patterns. Omega 2005;33:333–43.

[8] Bourgeois LJ, Eisenhardt KM. Strategic decision processes in high velocity environments: four cases in the microcomputer industry. Management Science 1988;34(7):816–35.

[9] Eisenhardt KM. Making fast strategic decisions in high-velocity environments. Academy of Management Journal 1989;32(3):543–76.

[10] Schlimmer JC, Fisher D. A case study of incremental concept induction. Proceedings of the 5th National Conference on Artificial Intelligence, 1986. p. 496–501.

[11] Utgoff PE. ID5: an incremental ID3. Proceedings of the 5th International Conference on Machine Learning, 1988. p. 107–20.

[12] Utgoff PE. Incremental induction of decision trees. Machine Learning 1989;4:161–86.

[13] Cheung DW, Han J, Ng VT, Wong CY. Maintenance of discovered association rules in large databases: an incremental updating technique. Proceedings of the 1996 International Conference on Data Engineering, 1996. p. 106–14.

[14] Thomas S, Bodagala S, Alsabti K, Ranka S. An efficient algorithm for the incremental updation of association rules in large databases. Proceedings of the 1997 ACM SIGKDD, 1997. p. 263–6.

[15] Can F. Incremental clustering for dynamic information processing. ACM Transactions on Information Systems 1993;11(2):143–64.
[16] Domingos P, Hulten G. Mining high-speed data streams. Proceedings of the 2000 ACM SIGKDD, 2000. p. 71–80.
[17] Rastogi R. Guest editor introduction: special section on online analysis and querying of continuous data streams. IEEE Transactions on Knowledge and Data Engineering 2003;15(3):513–4.
[18] Hoeffding W. Probability inequalities for sums of bounded random variables. Journal of the American Statistical Associations 1963;58:13–30.
[19] Hulten G, Domingos P, Spencer L. Mining time-changing data streams. Proceedings of the 2001 ACM SIGKDD, 2001. p. 97–106.
[20] Manku GS, Motwani R. Approximate frequency counts over data streams. Proceedings of the 28th VLDB Conference, 2002. p. 335–45.
[21] Zhang T, Ramakrishnan R, Livny M. BIRCH: an efficient data clustering method for very large databases. Proceedings of the 1996 ACM SIGMOD, 1996. p. 103–14.
[22] Guha S, Mishra N, Motwani R, O'Callaghan L. Clustering data streams. Proceedings of the 2000 Annual Symposium on Foundations of Computer Science, 2000. p. 1–8.
[23] Guha S, Mishra N, Motwani R, O'Callaghan L. Clustering data streams: theory and practice. IEEE Transactions on Knowledge and Data Engineering 2003;15(3):515–28.
[24] Brin S, Motwani R, Ullman JD, Tsur S. Dynamic itemset counting and implication rules for market basket data. Proceedings of the 1997 ACM SIGMOD, 1997. p. 255–64.
[25] Kleinberg J, Papadimitriou C, Raghavan P. A microeconomic view of data mining. Journal of Data Mining and Knowledge Discovery 1998;2:311–24.
[26] Sliberschatz A, Tuzhilin A. What makes patterns interesting in knowledge discovery systems. IEEE Transactions on Knowledge and Data Engineering 1996;8(6):970–4.
[27] Gibson G, Meter R. Network attached storage architecture. Communications of the ACM 2000;43(11):37–45.
[28] Ganti V, Gehrke J, Ramakrishinan R, Loh WY, A framework for measuring changes in data characteristics. Proceedings of the 18th ACM SIGMOD Symposium on Principles of Database Systems, 1999. p. 126–37.
[29] Mannino MV, Mookerjee VS. Optimizing expert systems: heuristics for efficiently generating low-cost information acquisition strategies. INFORMS Journal on Computing 1999;11(3):278–91.
[30] Moore JC, Whinston AB. A model of decision-making with sequential information-acquisition (Part 1). Decision Support Systems 1986;2(4):285–307.
[31] Moore JC, Whinston AB. A model of decision-making with sequential information-acquisition (Part 2). Decision Support Systems 1987;3(1):47–72.
[32] Elkan C. The foundations of cost-sensitive learning. Proceedings of the 2001 International Joint Conference on Artificial Intelligence, 2001. p. 973–78.
[33] Elovici Y, Braha D. Decision-theoretic approach to data mining. IEEE Transactions on Systems, Man and Cybernetics Part A 2003;33(1):42–51.
[34] Mookerjee VS, Dos Santos BL. Inductive expert system design: maximizing system value. Information System Research 1993;4(2):111–40.
[35] Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering 2005;17(6):734–49.
[36] Schafer JB, Konstan J, Riedl J. Electronic commerce recommender applications. Data Mining and Knowledge Discovery 2001;5(1–2):115–52.
[37] Ganti V, Gehrke J, Ramakrishinan R. Mining very large databases. IEEE Computer 1999;32(8):38–45.
[38] Linden G, Smith B, York J. Amazon.com recommendations. IEEE Internet Computing 2003;7(1):76–80.
[39] Bellman R, Dreyfus SE. Applied dynamic programming. Princeton, NJ: Princeton University Press; 1962.