ABSTRACT

In the last decade, researchers have recognized the need of an increased attention to a type of knowledge discovery applications where the data analyzed is not finite, but streams into the system continuously and endlessly. Data streams are ubiquitous, entering almost every area of modern life. As a result, processing and learning from multiple data streams have become important and challenging tasks for the data mining, database and machine learning communities. Although a substantial body of algorithms for processing and learning from data streams has been developed, most of the work is focused on one-dimensional numerical data streams (time series) or a single multi-dimensional data stream. Only few of the existing solutions consider the most realistic scenario where data can be incomplete, correlated with other streams of information and can arrive from multiple heterogeneous sources.

This paper discusses the requirements and the difficulties for learning from multiple multi-dimensional data streams inter-linked according to a pre-defined semantic schema (multi-relational data streams). The main research problem is to develop a time-efficient, resource-aware methodology for linking and exploring the information which is arriving independently and in an asynchronous way from its respective sources. The resulting framework has to enable, at any time error-bounded approximate answers to aggregate queries commonly issued in the process of multi-relational data mining. In particular we focus on the task of learning regression trees and their variants (model trees, option trees, multi-target trees) from multiple correlated streaming sources. To the best of our knowledge, no other work has previously addressed the problem of learning regression trees from multi-relational data streams.

Keywords

Regression trees, multi-relational data streams, any-time learning, one-pass approximate summaries

1. INTRODUCTION

Most of the existing algorithms for learning on data streams are designed for data represented in the standard attribute-value (propositional) format, i.e., a stream of tuples continuously populating a single data table. However, in most of the real-world and real-time knowledge discovery applications the data is structured and is usually represented by multiple correlated data streams. Consider as an example a real-time financial analysis problem, where one needs to take into account a fast stream of transactions and a stream of reclaims, attached to a much slower stream of customers (new and recurring ones) and a table of products which is mostly static over longer periods of time.

It is not difficult to see that, in order to properly analyze real-world data, it is necessary to take into account all the information available, as well as the relations that naturally exist among the data entities (e.g., the information on user purchases, navigation habits, personal data, relation to other users and so on). The enriched structure of the input can greatly leverage the ability to extract useful knowledge, but makes the learning task highly challenging. The increased complexity of the input space results in an explosion of possible hypotheses that a learning algorithm needs to examine.

The data stream setting by itself raises a number of issues which are not easily solved by traditional machine learning and data mining algorithms, such as the one-pass requirement for processing each learning example using a constant processing time and limited memory. A straightforward adaptation of incremental learning is not an effective solution to the problem. Algorithms must be computationally efficient, resource aware, adaptive and robust. To successfully deal with the evolving nature of data streams, models need to be continuously monitored and updated in real-time.

The biggest challenge however is linking the information which is streaming from several different sources. The difficulty of the task is mainly due to the fact that streams have different update speeds and are most often not synchronized. Most of the existing work assumes that all the information required for learning (defined by the structure of the input) arrives simultaneously, or employs computationally heavy methods for propositionalization over windows of most recent facts. We argue that the existing approaches do not consider all the aspects of the problem or are unable to leverage the power of multi-relational learning on data
streams. In that light, we formulate the problem differently and motivate for a new approach which will enable a more powerful, real-time and resource efficient analysis.

In the next section, we give several motivating examples. We define the problem setup and identify some of the emerging research questions. Next, we proceed with a short survey of the existing solutions, and discuss possible extensions and our preliminary ideas. We further give a sketch of the research plan guiding the thesis project and present our expected contributions. Finally, we conclude with a discussion.

2. EXAMPLE APPLICATIONS

The problem of learning from streams of relational objects is well motivated by many real-world applications. We will look at two motivating examples drawn from everyday life:

1. Lets take the example of a loan-analysis problem from the PKDD’99 challenge [21]. A banking database is given which consists of eight relations. Each relation describes the different characteristics of a client. For example, the Client relation contains a customer’s age, while the Account relation identifies a customer’s banking account information. In other words, each relation from this database provides different types of information for the concept to be learned, i.e. whether a customer is a risk or not. The target relation is Loan, and the target attribute is the status of a loan. Simultaneously with the approval of new loans, different banking transactions are being performed and banking orders are being issued, as well as new credit cards being given. Each of these activities are being queued in the corresponding relation, and represent a stream of information.

2. Another example is the city traffic and its analysis problem. Traffic events like accidents, congestions, road-blocks, etc. are the target objects of interest and are associated with additional streams of information such as sensor traffic measurements, news, GPS messages from the vehicle involved, weather data at the time of the event and many more, all recorded on a daily basis. A machine learning algorithm can be employed to learn a model for describing or predicting the duration of city traffic events (accidents, jams, etc.). Due to the number of accidents (and after-events) that happen on a daily basis, it is impractical to store every data point registered by a sensor for longer time than needed to update the underlying model. Thus, an incremental on-line approach is necessary. Although, in this example a pre-defined semantic schema is lacking, the streams are related on a spatio-temporal basis and as such are adherent to the proposed methodology for analysis.

3. Finally, we add-up to this list “favorable trading”, which refers to performing stock transactions that are favorable to the engaging party, i.e., selling before a stock plummets or buying before a stock goes up. In order to build classification models that identify patterns of “favorable trading”, all trading transactions and their outcomes must be examined. As discussed by Xu et al. [20], stock transactions are not isolated or independent events; they are related to many other data streams, e.g., phone calls between dealers and managers/staff of public companies. Thus, it is necessary to mine multiple related data streams. The stream of Transactions can be the target stream accompanied with an information on the goodness of each transaction, and related to the streams of Traders, Phone calls and Companies, each containing additional information on the transactions issued, the traders, their activity before and after trading, and information on the companies whose stocks were involved.

Following this line of reasoning one can think of many other real-world scenarios in which it is interesting and necessary to learn from multiple streaming sources of information. Learning over multiple correlated streams of facts is a new problem and requires rethinking the initial assumptions and a new mind-set. In the next sections, we describe the learning setup and formulate the research question.

3. PROBLEM FORMULATION

3.1 Relational Streams

From the relational database perspective the learning setup consists of a set of relations $R_1, R_2, \ldots, R_r$, each of which is representing one distinct multi-dimensional data stream. The learning algorithm is allowed to see the data tuples in $R_1, R_2, \ldots, R_r$ only once and in fixed order as they are streaming in from their respective sources. The order of tuple arrival for each relation $R_i$ is arbitrary. Further, the update frequency of each relation is different and unknown up front. The learning algorithm is also allowed a certain amount of memory, significantly smaller than the total size of of the data set(s). A similar stream processing architecture has been previously described [10, 14, 8, 11].

Beside the relations $R_1, R_2, \ldots, R_r$, usually referred as background relations, there is one additional relation which is considered as the target stream $R_t$. Each record in the target relation corresponds to a single relational object in the database. Each relation has at least one key attribute, either the primary key attribute and/or the foreign key attribute. Foreign key attributes link to key attributes in other tables. This link specifies a join (association) between two tables. Each entity in a given relation can be linked to multiple entities in another relation $R_i$, resulting in a subset of the original tuples from $R_i$ (let $T_i(R_i) = \{t_1, \ldots, t_j\}$ denote the bag of tuples in table $R_i$ that correspond to tuple $t$ in table $R_t$). Each tuple in the target relation $R_t$ includes a target attribute, whose values we would like to predict or correlate with the values of other attributes.

3.2 Multi-Relational Regression

A multi-relational regression problem deals with a regression task when data is distributed in multiple tables (relations) linked by a pre-defined set of associations.

Definition 1. The task of relational regression is to find a function $F$ which maps each tuple $X$ of the target relation
Thus, given the schema of a relational database the goal is to find a-priori unknown and interesting patterns, that can explain or predict the target attribute and that not only involve attribute-value descriptions but also structural information denoted by the associations between relations.

Regression trees can represent a mapping of real values to ranges and sets of values of attributes from the target and the background relations, respectively. They are easy to interpret by human non-experts and in many applications interpretation is just as important as predictive accuracy. They can handle mixed variable types with ease, and are very robust to irrelevant or redundant attributes.

Definition 2. A relational regression tree is a binary tree in which every internal node contains a test which is a conjunction of first order literals, and every leaf (terminal node) of the tree contains a real valued prediction. More precisely, the test corresponds to a conjunction query over the given database.

For easier comprehension, in Figure 1 is given an example of a relational classification tree (instead of a regression tree) constructed using a batch relational tree learning algorithm TILDE [4] using the PKDD’99 challenge dataset [21]. The tree gives a mapping of the status of a loan application (positive or negative) to attribute values from the relations Transaction, Card, Loan and Order. Each decision node of the tree contains a test on the value of a particular attribute or on the validity of a conjunction of first order literals. The leaf nodes hold the prediction (in this case a class value, colored in orange if positive and in green if negative). The leaf nodes additionally hold information on the number of positive and negative examples of loan applications covered the set of conditions.

In order to induce a regression or a classification tree from a given data set a generic learning algorithm follows a standard top-down approach in which at each step a decision node is added through a successive refinement procedure (improving the predictions for the examples in the training set). The choice of the refinement is guided by a suitable impurity measure (e.g., reduction of the variance of the target attribute). This refinement procedure continues until some termination condition is met. At that point, a leaf node with a prediction is introduced. The prediction is computed as the mean value $\bar{y}$ for all the tuples that satisfy the conjunction of conditions in the decision nodes on the path from the root of the tree to the corresponding leaf.

Each path from the root node till a leaf represents a conjunction of first order literals or a relational pattern. If we traverse the tree given in Figure 1 starting from the root node and always following the left branch (positive outcome of the test) we will obtain the following relational pattern:

\[
?-\text{Transaction}(X, \ldots, \text{C}), \text{equal}(C, \text{val_char}), \text{Card}(X, \ldots, \text{C}), \text{Loan}(X, \ldots, \text{C}).
\]

Thus, a relational pattern is a complex selection operation over the semi-join of the target relation and some or all of the background relations.

To perform incremental learning of decision trees (for classification or regression tasks), the basic idea is to make an assumption that only an i.i.d. sample of the learning examples is enough the make a stable refinement decision. If the assumption holds, each refinement step can be performed after observing a number of examples from the stream. The refinements of the children nodes will be decided on the next succeeding examples which will be rooted according to the conditions in the internal nodes. Several propositional online algorithms [19, 20] based on this same idea have already been developed and shown to perform comparably in terms of accuracy with their batch counterparts, while achieving learning speeds in an order of magnitude.

3.3 Query Refinements

The refinement procedure greedily considers every possible refinement that can be made to the current relational pattern with respect to the examples seen and selects the optimal refinement (i.e., the one that maximally reduces the variance of the prediction). The possible set of refinements is governed by the language bias, and the set of associations among the relations.

A refinement of a node basically translates to an addition of a new literal that contains a condition on some attribute on the involved relations or a new variable. In the second case, such a refinement requires a join with one of the background relations. For example, a possible refinement of the relational pattern:

\[
?-\text{Transaction}(X, \ldots, \text{B}, \ldots), \text{Loan}(X, \ldots, \ldots).
\]

is a conjunction with the literal equal(B, val_balance), which will filter-out only those loan applications for which there exist a transaction issued from the same account number and with a value for the attribute B equal to val_balance. If we wish to evaluate a refinement using the literal Card(X, ...), we will need to perform a join with the relation Card, which will give the loan applications with account numbers for which a credit card has been approved.

To perform a refinement of a given node we need to calculate some statistical measures for the quality of the refinement, like the support of the relational pattern on the resulting data partitions. The following query will calculate the number of relational objects that satisfy the joint conditions represented by the above mentioned relational pattern:

\[
\begin{align*}
\text{SELECT COUNT(DISTINCT Loan.foreign_key)} & \quad \text{FROM Loan} \; l, \text{Transaction} \; t \\
\text{WHERE} \; l.foreign_key = t.foreign_key & \quad \text{AND} \; t.balance = \text{val_balance}
\end{align*}
\]

The main difficulty lies in the fact that tuples in the distinct relations do not arrive simultaneously, and cannot be
Figure 1: A relational classification tree for the PKDD’99 challenge dataset.

linked directly for estimation of the query refinements. For example, in order to count the number of positive loan applications approved on a given account X, for which at least one withdrawal transaction to another bank has been recorded, it is necessary to link the tuples from the target relation Loan and the background relation Transaction and select only those for which the chosen conditions are satisfied:

?- Loan(X, pos), Transaction(X, withdrawal, to_other_bank, _).

However, the timestamps of the loan applications and the transactions need not be aligned. A user which has been approved a loan might not perform a transaction for a very long time, and vice versa. Thus, either all the tuples from the relation Transaction need to be stored or the joint appearance of all the possible conditions to be marked in a summary data structure. In both of the cases, the amount of information we will be able to store depends on the algorithm’s available memory and the per-item-processing time. Namely, if the updates of the relations are performed at rates higher than the processing abilities of the algorithm, part of the information will be lost.

4. RELATED WORK

Three main directions has been distinguished in the area of relational data mining: 1) Inductive Logic Programming (ILP); 2) First order extensions of probabilistic models; and 3) Approaches that borrow ILP techniques but define a search space that consists only of database queries. We will consider only the first and the third approach, and only in the context of incremental learning.

The first subsection outlines the recent work which motivates extending the first order reasoning framework on data streams, while the second subsections surveys existing algorithms for learning classification trees on multi-relational data streams, mainly from a relational database point of view.

4.1 Inductive Logic Programming

One of the main logical approaches which has been used to scale-up ILP algorithms is the idea of learning from interpretations where the locality assumption is exploited [5]. In learning from interpretations each example ε is represented by a separate Prolog program encoding its specific properties as sets of facts defining an interpretation and the background knowledge is given in the form of another Prolog program. As discussed in the work of Dries and De Raedt [12], this setting is appropriate for read-once stream mining since each learning example represents a small individual database. In their work they show some preliminary results on upgrading the clausal discovery paradigm towards learning on streams.

Another work that explores the read-once learning from interpretations is the work of Driessens et al. [13] proposing an incremental first order decision tree learner for speeding up relational reinforcement learning. The proposed algorithm TG is based on the incremental propositional learning algorithm G [7] that updates its theory incrementally as new examples are added. An important feature of the algorithm as stated, is that examples can be discarded after they are being processed. The algorithm stores only the current decision tree, and for each leaf node statistics for all tests that could be used to further split that leaf.

The main disadvantage of both of these approaches is that they assume a stream of interpretations readily available for learning. This assumption is not a realistic one, since the set of facts which describe the learning example cannot be easily decided in-front. As discussed in the above described example, some of the facts for a given loan application might arrive much later than expected.
4.2 Multi-Relational Data Mining

Real-world data usually resides in relational databases, thus, a common practice has been to convert the data to a format acceptable by the ILP system. Due to the existence of different ILP engines in terms of the input specification, the use of ILP algorithms in relational data mining solutions has been limited. Thus, most of the proposed solutions for relational learning on streams are from the relational database perspective, and mostly for classification and clustering tasks [26, 22, 23, 24, 18]. Among them, we will focus only on the classification algorithms.

To decide on the amount of information which will be used for learning in both of the proposed solutions it is assumed that streams are synchronized, i.e. the new tuples are related with each other more than the old ones. Every stream has a sliding window of fixed size pre-defined by the user.

In the work of Siddiqui and Spiliopoulou [24] the sliding window is combined with a sampling procedure that enables the algorithm to cope with the high speed and the continuous feeds of data. All the tuples included in the windows are joined in a preprocessing step known as propositionalization. Additional aggregation steps are further employed to deal with one-to-many types of associations when bag of tuples corresponds to a single target tuple. Their work extends the CVFDT algorithm [19] which is a propositional algorithm for learning classification trees over time-changing data streams to deal with a special type of concept drift, defined over the relational objects.

Propositionalization has been previously used in batch multi-relational learning as a method that enables utilization of of-the-shelf advanced machine learning methods designed for single-table data (propositional learners). However, it has some disadvantages as reported by several authors [27, 17]. Namely, "flattening" the relations is a non-trivial task and might require extensive preprocessing efforts. In addition, the resulting flat file may contain many NULL values, or result in a very big table with exponentially large number of additional attributes which causes further scaling and over-fitting problems for propositional algorithms [27]. The shortcomings from the propositionalization of relational data have led to the development of many relational algorithms which are able to directly operate on relational databases.

The paper of Hou et al. [18] on the other hand proposes an algorithm called RedTrees for learning relational decision trees from data streams without propositionalization. To reduce the high complexity of the join process the authors propose a sampling strategy for selecting a minimal subset of target keys (target objects) that will still enable to perform splitting decisions with high confidence. The sampling strategy is based on the periodical sampling probability bound which is a variant of the Chernoff probability bound where both the error ε and the confidence δ are set as user-defined input parameters.

After the initial \(\frac{1}{4δε^2}\) target keys are seen and stored along with the corresponding related facts from the other streams, they are fed to a batch learning algorithm which uses a Prolog engine and builds a relational tree. The next target keys are used for maintaining the tree, such that the nodes whose confidence is evaluated to be less than a threshold will be pruned periodically, while leaves would be expanded. The sampling probability bound is used to decide on the amount of facts which will be required to perform the maintenance operations.

The expanding procedure requires the learning examples to be stored in the leaf and joined when necessary. Thus, to evaluate the refinements of the relational patterns explicit join operations need to be performed which jeopardize the speed of learning. For that reason, their solution is not efficient and was not shown to provide any significant improvements in speed or accuracy over the batch algorithm for learning first order decision trees TILDE[4].

5. PRELIMINARY IDEAS

In order to learn from multi-relational data streams we identified two general approaches which can be combined into an effective solution to the problem of linking the distributed information. Each is discussed separately in the following subsections.

5.1 Incrementally Built Interpretations

The first approach considers extending ILP algorithms designed for incremental relational learning on data streams. In the incremental setup for learning from interpretations, at each time step the ILP algorithm is given a set of interconnected facts, each of which belongs to a certain learning example. The main problem lies in defining the set of facts which are related to a given learning example. As discussed previously, the relevant information does not arrive simultaneously across all the streams and thus it is not possible to form an interpretation for learning without partially storing the streams of facts.

In theory, initially observed facts might become relevant after a very long time. Since it is unreasonable to store all the data up until that moment, one has to assume that very old records are not so important and the focus will be only on the most recent ones. Thus, as in the existing approaches the main idea is to define sliding windows of different size over each of the streams. The set of facts enclosed in the current windows will be used to create the learning examples in the form of interpretations. Each learning example will be processed only once without storing. The processing operation will include updating of the sufficient statistics required to evaluate a node refinement.

The decision on the sizes of the windows is crucial and defines the amount of information the learner will be able to see for each learning example. Since it is not possible to know in advance how much time the engine should wait until it has all the relevant facts for learning, this decision should be data driven and flexible. Most of the existing solutions use windows with fixed size or some heuristic to guide the process of adapting the window sizes. Both of these solutions provide no guarantees or theoretical support in terms of error bounds on the values of the evaluation function used in the query refinement procedure.

The problem of deciding the sizes of the sliding windows is basically a special case of the more difficult problem of sam-
Sampling records (facts) from the individual relations for the task of obtaining approximate answers on arbitrary distinct values queries. Gibbons [15] proposes an approach for distinct sampling, that collects a sample over the distinct values of the input in a single scan of the data. The obtained sample is guaranteed to provide highly-accurate estimates of the number of distinct values. In the same time, the samples can be incrementally maintained up-to-date in the presence of data insertions and deletions, with minimal time and memory overheads. However, this sampling technique if applied over joins suffers from the same difficulties as uniform sampling over joins [1]. Thus, the learner needs to either store the records for which additional information is expected and learn when the data becomes available, or try to summarize the discriminative combinations of conditions (relational patterns) while facts are streaming in without storing all the tuples.

5.2 Approximate One-Pass Summaries

The main idea behind the second approach is inspired by the efficiency of several existing methods for obtaining summary data structured for approximate answering arbitrary aggregate queries over multiple streaming relations [1, 16, 10, 14, 11, 28]. The procedure is known commonly known as sketching, because it results in a set of sketches obtained efficiently over the distinct data streams, which can be used to directly approximate different quantities of interest ($L_1$ and $L_2$ norms of vectors, number of distinct items in a sequence, join and self-join sizes of relations, item and range-sum queries). They have also been used to compute more sophisticated aggregates like quantiles, wavelets, histograms, and multi-way join sizes. For a more detailed review on the sketching techniques, please refer to the work of Cormode and Muthukrishnan [9].

In general, sketches are based on randomizing techniques that compute small, pseudo-random sketch summaries of the data. The basic sketching technique was introduced by Alon, Matias and Szegedy [2] and has been generalized for the more complex tasks of estimating counts, sums and averages of a subset of tuples satisfying a given conjunction of $n$ equi-join constraints of the form $R_i . A_j = R_k . A_l$ (where $R_i . A_j$ denotes the $j^{th}$ attribute of relation $R_k$). Another popular sketching technique is the Count-min sketch [9] which can be used for a variety of applications where accurate estimates of counts are required. In particular, it has been used for estimating entropy over data streams [3] which is the basic impurity measure typically used in any classification algorithm.

All of the above discussed techniques enable time and memory efficient summarization of data streams for the estimation of different quantities. However, they are not effective in capturing the join-based correlations among the attribute values from the different relations. This is because they summarize each attribute domain separately loosing the connections between the records. This has been the main observation in the work of Spiegel and Polyzotis [25] which propose a different type of data summaries that enable accurate selectivity estimates for arbitrary complex relational queries. The TuG synopses are a schema-level data synopses which are able to compress the join information. Basically, the TuGs represent data partitions along with aggregate statistical information which is used to estimate the probability that a query expression is satisfied on the tuples in the partition. To reduce the storage of a TuG the authors propose a compression operation based on clustering. For efficient clustering over a high-dimensional space the TuG synopses use count-min sketches.

6. RESEARCH METHODOLOGY

Here we would like to give a sketch of the research plan for attacking the problem formulated in Section 3. The work of Spiegel and Polyzotis [25] brings together the two high level ideas outlined in this paper. The base idea is to use one-pass schema-level summaries which can be incrementally maintained under insertions and deletions in order to select or sample the tuples that should be kept in the sliding windows for improving the learned patterns (tree). This is related to data reduction techniques that seek to compress the data by using the correlations that exist among the attribute values and the relations. The similar tuples which can be clustered and represented with the centroid can be removed from the window, which will enable new ones to be included. Besides the savings in space the centroid will be used to perform linking efficiently. The main idea is to correlate the tuples with the centroid instead with their truly associated records. This approximation will inevitably introduce some error which will mainly depend on the clustering process.

The TuG synopses [25] are designed for static relational datasets with complex join relationships. In order to be used in summarizing relational data streams an efficient on-line updating procedure is the main requirement. Thus, solving this problem would be the first attacking point. Having an efficient summarization engine that performs in near real-time will enable feeding a diverse range of machine learning algorithms with inter-linked tuples or approximate answers for evaluating incremental query refinements. Querying the join summaries removes the need to access the original data or perform explicit join processing. Of course, such data summaries can require a lot of memory, but by noticing that not all of the combinations are equally informative, parts of the summaries can be pruned using the partially built models and the same greedy approach used for learning.

7. OUR CONTRIBUTIONS

The bottle-neck in all the algorithmic solutions designed for learning on multi-relational datasets with complex join relationships is the relational join process which is inevitable in any refinement operation. This is a very important issue in the design of any on-line relational learning algorithm expected to perform in real-time. The main contribution of our research will be an efficient solution to the problem of linking the records observed over multiple data streams for the task of on-line approximate aggregate query answering. To the best of our knowledge there is no prior work that solves this problem efficiently. The applicability of our ideas will be explored for the problem of on-line regression using single regression trees or ensembles and their variants.

An interesting remark is that, regression trees can be easily upgraded to describe and predict multiple targets simultaneously by using the predictive clustering framework [6]. This framework encapsulates methods which seek to con-
struct clusters of examples that are similar to each other in the output space and simultaneously associate a predictive model (classification or regression) with each constructed cluster. Multi-target trees are relevant for real-world problems in which users are interested in the values of several parameters of interest explained by using a single model. The methods can be also extended to predicting other types of structured outputs (e.g., hierarchies, graphs, etc.).

8. REFERENCES


