Data Mining: From procedural to declarative approaches

Hendrik BLOCKEEL

KU Leuven, Department of Computer Science &
Leiden Institute of Advanced Computer Science, Leiden University

hendrik.blockeel@cs.kuleuven.be

Received 12 November 2014

Abstract This article provides a viewpoint on the past and possible future development of data mining technology. On an introductory level, it provides some historical background to the development of data mining, sketches its relationship to other disciplines, and introduces a number of tasks that are typically considered data mining tasks. It next focuses on one particular aspect that may play a larger role in data mining, namely, declarativeness. Despite the fact that many different data mining tools have been developed, this variety still offers less flexibility to the user than desired. It also creates a problem of choice: which tool is most suitable for a given problem? Declarative data mining may provide a solution for this. In other domains of computer science, declarative languages have led to major leaps forward in technology. Early results show that in data mining, too, declarative approaches are feasible and may make the process easier, more flexible, more efficient, and more correct.

Keywords Data Mining, Inductive Databases, Inductive Query Languages, Declarative Modeling Languages, Constraint Solving.

§1 Introduction

Data mining has been a very active research area in computer science since the 1990s. It concerns the generation of useful knowledge from large amounts of (possibly complex) data. To some extent, it shares these goals with the related fields of machine learning and statistics. However, in data mining, there is typically more focus on computational issues, such as scalability and efficient data access, and many methods are oriented more towards producing sets of results ("patterns"), rather than a single result (a "model").

In over two decades of data mining research, many different types of tasks have been considered, and many techniques proposed. As a result, a plethora of different methods and approaches now exist, many of which are based on complex mathematics or sophisticated algorithms. The focus in data mining has largely been on the development of methods, more than on understanding the relationships between them, even though important advances have been made regarding the latter too.

In this situation, it is increasingly challenging for practitioners to select, among the many approaches that are available, the one that best fits the problem at hand. They have to select first an algorithm, and then a parametrization for the algorithm, such that optimal results are obtained. In practice, their choice is limited by the tools and methods they happen to be familiar with, and therefore likely suboptimal.

Related to this is the problem of interpretation. It is well known that statistical methods are often used incorrectly, leading to incorrect conclusions; the field of machine learning has also suffered from this.[12, 13)] In data mining, too, using the wrong method and interpreting its results may lead to false conclusions.

The solution to both problems is in making data analysis more declarative. It should be possible for the user to describe the data analysis *problem*, rather than having to describe a method for *solving* it. Compare this to how SQL is used in database technology: SQL made it possible for users to set up and query databases in a much simpler manner than before. The user can ask complex questions without having to know any details about the complex retrieval procedures that are needed to answer that query. This has not only made it much easier to use databases, it has also led to better efficiency (the system chooses the execution strategy that it thinks is most efficient, rather than leaving this to the user) and made the process less error-prone (manually programming complex retrieval procedures is bound to lead to bugs). The ultimate goal of research on declarative data mining is to similarly change the way users are analyzing data.

Figure 1 visualizes the difference between "procedural" and "declarative" data analysis. The point of declarative data analysis is to maximally move

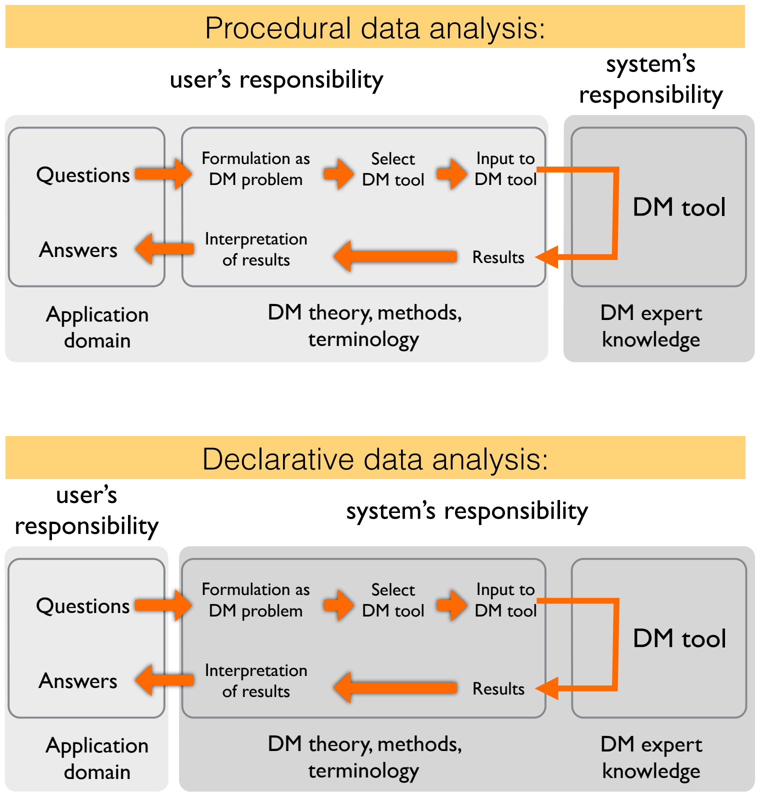

Fig. 1 Procedural and declarative data analysis. In the procedural approach, currently the state of the art, the user is responsible for translating a problem in the application domain to data mining terminology, mapping it to a standard data mining problem, selecting a suitable data mining tool, running the tool, and interpreting the results in the context of the application domain. In the declarative approach, much of this is done automatically by the system.

responsibilities from the user to the system, so that correctness and optimality can be guaranteed by the system, rather than relying on the user for this.

Achieving this goal requires research on several points. First, a language is needed in which a wide variety of data mining problems can be expressed. Ideally, such a language should allow the user to formulate questions in the user's domain language, rather than using specific data mining terminology. Second, state-of-the-art knowledge about correct data mining methodology must be built into the system. Third, efficient execution strategies must be developed, as well as methods for mapping data mining queries onto execution strategies.

There currently is no single research domain called declarative data analysis, but all of the above challenges are touched upon, often implicitly, in a variety of different research areas. This includes inductive databases, inductive query languages, constraint-based data mining, declarative modeling languages, constraint programming, statistics, probabilistic inference, and machine learning. Combining research results from all these areas may significantly advance the state of the art in declarative data analysis, and may in the long run radically change the way in which data analysis is conducted.

This article starts with a basic introduction to data mining, and gradually drills deeper into the topic of declarative data analysis. Section 2 sketches the broader context in which data mining is set, and provides some historical perspective. Section 3 focuses specifically on data mining. It familiarizes the reader with concepts, tasks and methods that will be referred to later on in the text, and further motivates the concept of declarative data mining. Section 4 discusses inductive databases and inductive query languages, which can be seen as a first step towards declarative data mining. Section 5 introduces the idea of using more powerful modeling languages for data mining, using a case study. Section 6 discusses the concept of "declarative statistics", showing that the utility of declarative approaches extends well beyond data mining in the narrow sense. Section 7, finally, presents concluding remarks.

§2 Data Analysis

The broader context in which data mining is set, is that of data analysis. Data analysis is as old as science itself. Indeed, long before scientific theories were based on more fundamental models of nature, such as those provided by physics or chemistry, humans have built specific models from observations. A classic example is Kepler's laws, which were based on observation of the motion of celestial bodies, and which could be explained from more fundamental models only much later, using Newton's theory of gravitation. There are many more such examples in the natural sciences. Even in mathematics, which is inherently less empirical, patterns are often observed, and sometimes conjectured to hold in general, well before they are proven.

For ages, data analysis was hardly considered a separate discipline. Observations were made, and next analyzed, in whatever manner seemed appropriate. Probably the first separate field to study data analysis as a task in itself, was that of statistics. While some methods that we now consider statistical methods, such as least squares regression, were invented much earlier, statistics as a standalone scientific discipline originated only around the end of the 19th century.

Statistics has long focused on methods that have a sound mathematical basis and require relatively little computational effort. The invention of computers, mid twentieth century, made it possible to use more computationally intensive methods. But it also had another effect: large-scale collection and storage of data became ever cheaper, and the data to be analyzed were no longer restricted to numerical, ordinal or categorical data: much more complex forms of data, such as text, images, sound, video, and networks, could be stored, and interest grew in analyzing such data, too. This development gave rise to the fields of machine learning, and later data mining.

Machine learning originated as a subfield of artificial intelligence in the 1980s. As the name suggests, it was generally concerned with learning something new (models, theories) from existing knowledge. Examples are: learning the definition of some concept from examples (concept learning), learning to solve a task by observing how others solve it (behavioral cloning) or by exploration (reinforcement learning), mimicking biological learning (neural networks), etc. With this variety of tasks came a variety of input and output formats used by the learner.

The term *data mining* became used in the 1990s to refer to a field that built on earlier work in statistics, machine learning, and databases, but which focused in addition on challenges related to scalability, data complexity, interfacing with relational databases, exploratory data mining, visualization, and more. Data mining also introduced a number of tasks, such as association rule discovery, that had not been considered before. Typical for these tasks is that they are set-oriented: they are of the form "find all patterns that fulfill certain criteria". As such, they are more reminiscent of typical database queries.

Although the goals of statistics, machine learning and data mining are different, they share the inductive reasoning aspect: generalizing from a sample towards a broader population. As such, it is no surprise that, for instance, decision tree learning was developed independently in statistics [7] and machine learning,[26, 27], and later adopted and refined by the data mining community.[14] Similarly, approaches such as support vector machines, probabilistic models, etc., are nowadays considered to belong to all three disciplines.

More recently, the terms "big data" and "data science" have entered the scene. As data mining already focused on handling large amounts of data, the term *big data* is sometimes used as a synonym for it. But it is more than that: it covers not only the analysis of large sets of data, but everything related to its processing, from collection to exploitation. Indeed, when a telescope generates terabytes of data per day that are to be analyzed, even the storage of the data itself, or its transportation over a network, becomes challenging. These challenges can reasonably be considered to lie outside data mining; they are more relevant for databases and distributed systems.

Finally, the term *data science* has become popular from 2010. The term has been claimed by statisticians, machine learners and data miners alike. This author's viewpoint is that data science actually covers all these areas, including big data, and is an excellent umbrella term for the variety of methods, tools, and theory that have been developed. Consider again the example of the telescopes: An astronomer who wants to analyze the data produced by a telescope does not care whether the methods used for analyzing these data are categorized under statistics, machine learning, data mining, or whatever other denomination; neither does she care in what area of computer science the methods for storing and transporting these data have been developed. The ultimate goal is simply to turn the data into knowledge. Data science is naturally defined as the science that covers all aspects of this challenge.

§3 Data Mining

3.1 Data Mining and Knowledge Discovery

Data Mining is often defined as one particular phase in a more general process called knowledge discovery. Frawley et al.[16)] define knowledge discovery as follows: *Knowledge discovery is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data.* The discovered knowledge is assumed to be in the form of a *pattern*, which is defined as a statement that describes relationships among a subset of the data.

The whole knowledge discovery process comprises multiple steps, of which data mining is typically considered one step, namely the one where a set of data is analyzed and patterns are extracted. Figure 2 provides an overview. The first step in the process comprises the *collection* of the data. In some contexts, data collection is automated and highly reliable, so that errors or missing data are rare; in other contexts, it is only semi-automatic and possibly unreliable, and errors or missing data may be abundant. In the latter case, a subsequent

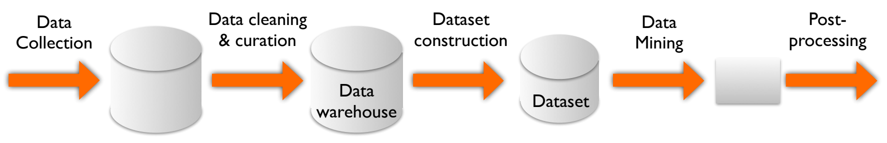

Fig. 2 The knowledge discovery process

step of *data cleaning*, or sometimes *data curation*, may be necessary to improve the quality of the data and guarantee its validity.

The cleaned and/or curated data are often stored in a so-called data warehouse: this is a database system tuned towards storage, retrieval and analysis of large amounts of data.

The stored data are typically preprocessed before being analyzed. This preprocessing often includes feature selection and feature construction: The data are transformed into a tabular format (sometimes called the standard format) where each data element is represented as one row in a table, and each column represents one feature or attribute that is thought to be potentially relevant. Many data analysis methods expect data in this format. However, it is not always possible to store all the information in a relational database into one table. The relationships between different data elements that are indicated in a relational databases may be such that it is impossible to store all the information relevant to a single individual in a single tuple in a table. We can therefore distinguish tabular data mining methods, which analyze data in the standard format, from relational data mining methods, which can directly analyze data in a relational database.[28)]

The actual data mining step consists of choosing some data mining algorithm, running it on the data, and storing the results. The results themselves may not be in a format easily digested by humans; therefore, a postprocessing step is typically added. Such postprocessing may include selection of the most relevant results, ranking, graphical visualization, and so on.

The above description covers many instances of the knowledge discovery process, but not all. Sometimes, data are produced at such a high rate that they cannot be stored; a stream of data is produced on a continuous basis, and each data element must be consumed right after it has been produced. Data mining in this setting is also called data *stream mining*. Whereas regular data mining methods have random access to the data (they can in principle look up any data element at the time they need it), data stream mining methods can look at each data element just once, for a brief time, before it is gone forever. In the knowledge discovery process, there is then no construction of intermediate objects such as data warehouses or datasets; the only object constructed is the model (the result of the data mining process), and this model is updated as new data arrives.

3.2 Typical data mining tasks and approaches

As said, data mining is closely related to machine learning and statistics, and there are no strict boundaries between these areas. However, some tasks and methods have been studied mostly in the data mining research community, and as such can be considered typical data mining tasks. A few of them are listed here.

Association rule discovery. Assume there is a set of items I and a set of transactions T, and with each transaction corresponds a subset of the items in I. The frequency of an itemset S, denote $f(S)$, is the number of transactions in T that are a superset of S. An itemset is called frequent if its frequency is above some given threshold. An association rule is of the form $X \rightarrow Y$ where X and Y are itemsets. Its support is defined as $f(X \cup Y)/|T|$, and its confidence is defined as $f(X \cup Y)/f(X)$. The task of association rule discovery is typically defined as follows: given I and T, find all association rules whose support and confidence are above some given threshold.

The classic example of association rule discovery is market basket analysis. A transaction represents a sale in some supermarket; its items are the products bought. If the association rule `milk, bread` $\rightarrow$ `cheese` has a confidence of 0.8 and a support of 0.2, this means that 80% of all people who buy milk and bread also buy cheese, and this rule describes a relatively large percentage (20%) of all customers. Such rules can give insight in customer purchases.

Many systems for association rule discovery first discover all frequent itemsets, and in a second step derive the association rules from these. This two-step process was introduced in the seminal work on the Apriori algorithm.[2)] It has allowed researchers to study the first step, frequent itemset discovery, as a problem in itself, and to generalize it towards frequent pattern discovery.

Frequent pattern discovery. This task is similar to frequent itemset discovery, but focuses on more general kinds of patterns than itemsets. For instance, each transaction may be a graph, rather than a set of items, and the task is to find frequent subgraphs. The frequency of a graph G is here defined as the number of transactions that contain a subgraph isomorphic to G; G is frequent if its frequency is above some threshold. In a variant of this problem, there are no separate transactions: there is just one large graph H, and the frequency of G is, roughly, the number of subgraphs of H that are isomorphic to G. A precise definition of frequency is more difficult to give, in this case; for instance, one needs to decide whether multiple occurrences of the pattern are allowed to overlap.[33)] Discovery of frequent subgraphs is relevant, for instance, in pharmacology (analyzing molecular structures) or social network analysis.

Predictive modeling. Predictive modeling is a task also considered in machine learning and statistics. It consists of learning a function that can be used to predict the value of some property, called the "target" attribute. Usually, a set of example (x, y) pairs is provided, where $x \in X$ and $y \in Y$ are tuples containing numerical or symbolic components, and the task is to learn a function f such that, for all examples, $f(x)$ is equal to y, or at least similar to it. A so-called loss function $l(y_1, y_2)$ defined over $\mathcal{Y} \times \mathcal{Y}$ expresses how similar y_1 is to y_2. Often, the data set D is considered to have been drawn randomly from some distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, and the task is to find the function f that does not just minimize the loss over D, that is, $\displaystyle\sum_{(x,y)\in D} l(f(x), y)$, but the expected loss over the distribution $\mathcal{D}$. When $\mathcal{Y}$ is a set of symbolic values, these values are often called classes, or labels, and the task is called classification. When $\mathcal{Y}$ is numerical, the task is called regression. When $\mathcal{Y}$ contains vectors of numbers, it is sometimes called multivariate regression; when it contains tuples of booleans, the task is called multilabel classification;[30)] when it contains values with a more complex structure, it is called structured prediction.[29)]

Clustering. Clustering is a task that is ubiquitous in data mining, as well as in statistics or machine learning. The task is to identify structure in data, by finding groups of instances such that instances within a group are similar, and instances in different groups are dissimilar. Similarity is often expressed using a distance metric: two instances are similar if the distance between them is small. Many different types of clustering methods can be distinguished. Some clustering methods define clusters *extensionally* (by listing their elements), other methods define them *intensionally* (by describing properties of the cluster or its elements). Some methods return a *partition* (each subset in the partition is a cluster), others return a *hierarchical clustering* (clusters contain subclusters, which again contain subclusters, etc.); such a hierarchical clustering is usually visualized as a *dendrogram*. Clustering is usually considered an example of unsupervised learning: no information is provided as to which instances belong to which cluster; but semi-supervised versions exist, where, for instance, for some pairs of instances the system is told whether these instances should be in the same cluster or not.[10)]

3.3 Variety of tasks and approaches

The standard task definitions listed in the previous section do not always perfectly match the tasks a user is interested in.

For instance, one problem with association rule discovery is that it often yields too many association rules; many of these closely resemble each other, and many may be uninteresting to the user. Ideally, the user should be able to indicate more precisely what kind of rules he wants to see. E.g., the user may be interested only in association rules that contain a particular type of items in the head or body, in association rules that fulfill more complex grammatical rules (expressed, e.g., using a regular expression[17)]) or in association rules that fulfill other quantitative constraints than the ones based on confidence and support. A straightforward way to implement a system that looks for such rules, is to have it return all association rules fulfilling the weaker conditions (confidence and support) and then filter according to the remaining constraints. Given the number of rules that may pass the weak constraints, such an approach can still be very inefficient. Researchers have therefore searched for ways to "push the constraints into the mining process": the more constraints can be exploited during the search process, the more efficient it becomes. Each different type of constraint requires a different approach to pushing it into the search. This has led to the development of many different association rule discovery systems.

A similar situation exists in clustering. Many different clustering systems exist; they differ in terms of the goal (the type of clustering they try to define, and the criteria they try to optimize), and the way they work. In addition to this, some clustering systems can take constraints into account: these constraints may indicate that two instances should be in the same cluster (must-link constraints), or should be in different clusters (cannot-link constraints).[32)] Other forms of constraints are so-called chunklets, sets of instances that should all be in the same cluster;[4)] "example clusters", examples of clusters that one wishes to see in the clustering returned by the system;[20)] and more.[34)] Given that many different types of clustering systems exist, and for all of them it may be meaningful to extend them so that they can handle certain types of constraints, the already large number of existing clustering approaches can easily be multiplied by a relatively large factor, just to cater for all possible combinations of goal criteria and types of constraints.

Association rule discovery and clustering are not the only fields that have seen the development of many different systems, all solving a slightly different problem. More generically, a similar effect is visible in machine learning, which in recent years has shifted strongly towards statistical or Bayesian methods, which model the problem as a convex optimization problem and then run a solver. For many problems that are essentially small variations of each other, a new model is proposed and next solved. This modeling, however, is non-trivial, to the extent that it is usually presented as a novel approach to solving some problem.

Thus, given a particular problem, the user is confronted with the following tasks: find out whether this kind of problem has been studied before; if so, obtain and use an implementation solving that problem; if not, either develop a new solution for the problem, or find a system that can solve a closely related (but not identical) problem and use that system to approximately solve the problem at hand. None of all this is easy.

This situation begs the question whether it is possible to develop an approach in between these extremes; a single system or approach that is flexible enough to solve a variety of tasks, and yet easy enough to use. That is exactly what declarative data mining aims at.

3.4 Declarative data mining

Declarative approaches to machine learning and data mining have been around for some time. For instance, in inductive logic programming (ILP)[23, 24], the concept of declarative bias has always played an important role. ILP systems start out from a knowledge base expressed in a first-order logic based language such as Prolog, and inductively construct new knowledge (a logical theory) from this. The expressiveness of logical languages implies that a very wide range of learning tasks can be modeled in this way, and because of this, ILP systems need some guidance: the user needs to specify a "declarative language bias", a description of what type of results she is interested in. Much research has gone into effective ways of specifying the language bias and searching the corresponding hypothesis space. Note, however, that ILP systems offer flexibility in terms of defining the hypothesis space (the set of all objects among which solutions are to be found), but much less regarding the task itself, or the algorithm to be executed for solving that task.

Also the Bayesian approach to machine learning mentioned above, where users define an optimization problem to be solved, is to some extent declarative; however, as argued before, the translation from a problem description to a mathematical optimization problem is non-trivial and requires a lot of mathematical knowledge.

The idea of translating a data mining problem to a more generic type of problem, and then running a standard solver, is also present in a broad range of work where data mining tasks are expressed as constraint solving problems, and solved using constraint programming. It has been shown on several occasions that this approach can lead to very efficient solution strategies for (variants of) classic data mining tasks such as itemset mining or clustering.[25, 19, 9)]

In the remainder of this article, a number of approaches are explored that try to combine flexibility with ease of use. It should be stressed that this is by no means a complete overview. Topics have been selected with the goal of illustrating the variety of approaches and challenges involved in declarative data mining while at the same time minimizing the background needed to digest it.

§4 Inductive Databases and Query Languages

The concept of inductive databases was first proposed by Imielinski and Mannila.[21)] The basic idea behind it is the viewpoint that a dataset contains patterns, just like it contains data. That is, just like we can query a database for its content, we should be able to query it for patterns. Patterns should be "first class citizens": we should be able to query for them, store found patterns that we found interesting, etc., just like we can query and store any other database objects.

There are multiple challenges related to this. One is the development of a language that allows one to represent a broad class of models. Indeed, it is not difficult to develop a language for representing decision trees, neural networks, or association rules; but a language that can represent any kind of model is a different story. One attempt to do just that is PMML, which stands for Predictive Model Markup Language [18)]. It is an XML-based standard for representing predictive models. PMML was developed with the goal of simplifying the storage and exchange of a wide range of predictive models. For instance, an implementation of PMML in the statistical programming environment R provides functionality for exporting support vector machines, decision trees, (generalized) linear models, and more.

While PMML is relevant to the goals of inductive databases, research on it focused mostly on representation, and not on query languages for retrieving models or patterns from a database. Such query languages are also called inductive query languages. Within the inductive databases community, much of the research has focused on this problem. In the following, a number of such query languages are discussed. A more extensive discussion and overview of the state of the art is given by Džeroski et al.[15)]

4.1 Extensions of SQL for Data Mining

One approach towards developing inductive query languages is extending regular query languages, such as SQL, so that data mining queries can be asked. One of the earliest contributions in this direction was made by Meo et al.[22)], who introduced an extension of SQL that contains the MINE RULE operator. The operator makes it possible for a user to mine a database for association rules and store the result in a new table. Here are some examples of MINE RULE queries, taken from Meo et al.:

```
MINE RULE SimpleAssociations AS
SELECT DISTINCT 1..n item AS BODY, 1..1 item AS HEAD, SUPPORT, CONFIDENCE
FROM Purchase WHERE price <= 150
GROUP BY transaction
EXTRACTING RULES WITH SUPPORT: 0.1, CONFIDENCE: 0.2

MINE RULE OrderedItems AS
SELECT DISTINCT 1..n item AS BODY, 1..1 item AS HEAD, SUPPORT, CONFIDENCE
WHERE BODY.date < HEAD.date
FROM Purchase
GROUP BY customer
EXTRACTING RULES WITH SUPPORT: 0.1, CONFIDENCE: 0.2
```

The first query finds associations between products bought in the same transaction; this is the straightforward type of association rule, with one non-default constraint added, namely that we only want to find sets of items with a price not exceeding 150. The second query is more interesting, and demonstrates the power and flexibility of the query language: here, the user asks for associations among products bought by the same customer (not necessarily in the same transaction) where the items in the body were bought on an earlier date than the items in the head.

As a second example of how SQL can be extended towards data mining queries, consider the SCCQL language,[1] which allows the user to query a database for clusterings. The following example, taken from Adam et al.[1], illustrates how constraint-based clustering can easily be performed with such a language.

Consider a database that contains information on the evolution of cells in a cell colony. A colony starts out with one cell, which at some point divides into two; later on these cells again divide, and so on. A set of cells stemming from one original cell is called a Lineage. The table Lineage describes lineages, the table Cell describes individual cells, and the table Stateovertime describes the evolution of a cell over time; it contains for each cell a number of "snapshots" that describe, for instance, the length and width of the cell at the time the snapshot was taken. Some of the lineages start out with a mutant cell; the Lineage attribute Mutant indicates which mutation the cell had (0 means no mutation).

In this context, the query

```
CLUSTER LMean, WMean
FROM (SELECT c.Id, l.Mutant, AVG(s.Length) AS LMean, AVG(s.Width) AS WMean
      FROM Stateovertime s, Cell c, Lineage l
      WHERE l.ExperimentId=5 AND c.LineageId = l.Id AND s.CellId = c.Id
      GROUP BY c.id) AS Data
WITH SOFT MUST LINK WHERE Data.Mutant=0 BY Mutant
```

asks for clustering cells according to their mean length and mean weight (taken over their whole life span), imposing as a soft constraint that all non-mutant cells must be in the same cluster. Note that the creation of the table to be clustered (here called Data), and the specification of the constraints imposed on the clustering, are seamlessly integrated in the query.

4.2 Mining views

The Mining Views approach[6] takes the view of data mining as querying one step further. In this approach, the SQL language is not extended at all. Instead, views are defined on the database. These views contain objects such as itemsets, association rules, etc., and they can be queried just like any regular table. Thus, the possibility to mine for itemsets, association rules, decision trees, etc. is not provided by extending the language, but by extending the database structure, making use of language constructs that are standard in SQL.

As a proof of concept, the following approach was proposed and implemented.[6] For each table T, multiple views on that table are defined:

```
T_Concepts(Cid, A1, A2, ..., An)
T_Sets(Cid, Supp, Sz)
T_Rules(Rid, Cida, Cidc, Cid, Conf)
...
```

For a table T, the view T_Concepts contains all imaginable itemsets over T, that is, all subsets of the attribute set of T. It represents itemsets using the original attributes of T, with value `true` if the attribute is a member of the itemset and `?` if not (for the motivation of this choice, see the original article). Obviously, the table does not store all 2^n possible subsets. It is a virtual table; when queried, its contents are computed as the need arises. The table T_Sets contains the support and size of each itemset, and T_Rules contains all possible rules, indicating their antecedent, consequent, the concept that is the union of both, and the confidence of the rule.

To the user, asking for all association rules fulfilling some constraints looks exactly like querying a table containing all the association rules. For instance, the following query: [6)]

```
select Ante.*, Cons.*, R.Conf, S.Supp
from T_Sets S, T_Rules R, T_Concepts Ante, T_Concepts Cons
where R.Cid = S.Cid
and Ante.Cid = R.Cida
and Cons.Cid = R.Cidc
and S.Supp >= 3
and R.Conf >= 80
```

selects all association rules R that have a confidence of at least 80% and a support of at least 3,[*1] and reports for each rule the antecedent, consequent, confidence and support.

The following query returns all itemsets with an "area" (defined as support times size of the itemset) above 60, or support of at least 10:

```
select C.*, S.Supp, S.Sz, S.Supp * S.Sz as area
from T_Sets S, T_Concepts C
where C.Cid = S.Cid
and ( (S.Supp*S.Sz > 60) or S.Supp >= 10)
```

More specifically, the query returns for each such itemset the itemset itself (`C.*`), and its support, size and area.

Thus, data mining results can be obtained through simple SQL-based querying. Internally, a query to a "mining view" may of course cause the exe-

[*1] Different from the definition earlier in this paper, support is measured as an absolute number here, not as a percentage.

cution of an association rule discovery algorithm, but it is up to the system to decide how to answer the query; this is transparent to the user.

The user can construct new views on top of the existing ones. Thus, views corresponding to relatively complex data mining tasks can in principle be constructed. The database system can use any optimization strategies at its disposal to optimize the execution of these complex tasks. In practice, database systems contain highly advanced optimization strategies for "regular" queries but not necessarily for inductive queries. Little is currently known on how to optimize the latter.

§5 Modeling languages for data mining

The previous section made clear that inductive query languages can provide a lot of flexibility in terms of the data mining tasks that can be solved. However, in some cases, the expressive power of databases and query languages may not be sufficient to express the available knowledge about some domain and task. More powerful knowledge representation languages can then be used. The following case study shows the usefulness of such languages in data mining; it is based on original work by Andrews et al.[3] and Bruynooghe et al.[8]

The case study is set in the context of philology. A *written tradition* is a set of written documents that somehow relate to each other; typically, they are variants of the same story or text. These variants came into existence because in medieval times, books were copied manually by monks. In this process, variations got introduced into the new text, by mistake or otherwise. It could also happen that a monk started copying one text, and at some point switched to another text (for instance because the first had been damaged and was unreadable at some point), which resulted in a copy derived from more than one original; this condition is called contamination.

A *stemma* is a directed acyclic graph (DAG) that indicates which texts have been copied from which other texts. If no contamination is present, it is a tree. If the whole tradition stems from a single document, that document is the root of the DAG.

Some of the texts in a stemma may have been damaged or destroyed; for these texts, it may not be known which variant they contained for a particular phrase in the text. For a particular phrase, a document is called a *witness* if it is known which variant of the phrase it contained.

One task that philologists are interested in, is constructing a stemma from a set of documents; this task is quite similar to phylogenetic tree construction, an important problem in bioinformatics, and standard methods from bioinformatics are often used in this context. But in this particular case study, the philologists were interested in a different question: given a set of documents, some of which are witnesses for a particular phrase, and a stemma relating them to each other, is it possible that each variant of the phrase has a single point of origin (a single "source")?

If stemmata were trees, this question would be easy to answer, but for DAG-structured stemmata the problem is much more complex; in fact it is NP-complete.[8)] A special-purpose program was written to answer this question, but the program was complex and its correctness could not easily be proven.

A different approach was tried next: the problem was modeled using IDP3[11)], a modeling language that seamlessly integrates procedural and declarative components. An extension of first-order logic is used for the declarative components. The procedural component allows the user to state what tasks have be executed using the knowledge described in the declarative components.

In this particular case, the procedural component mostly takes care of reading files, printing output, etc.; most of this part is not relevant for the discussion here. The declarative component is more interesting. It specifies a theory, a declarative description of the problem to be solved. For the problem mentioned above, Figure 3 shows the theory as it is written in IDP3.

The knowledge base starts with defining a vocabulary. Essentially, it states the following:

- we will be talking about things called Manuscripts, and things called Variants
- there is a binary relationship CopiedBy among Manuscripts
- with each Manuscript is associated one Variant (called the "Variant In" that Manuscript)
- with each Variant is associated one Manuscript (called the "Source of" that Variant)

Next, a theory is provided that makes use of that vocabulary. The theory states that for all x, if x is not the source of its variant, then x must have been copied from a manuscript with the same variant. (The symbols `!` and `?` denote universal and existential quantification in IDP, `~=` denotes inequality.)

In the task we have here, the input will consist of a list of manuscripts and variants, a full specification of CopiedBy (which manuscripts are copied

```
/* --------- Knowledge base ------------------------ */

vocabulary V {
  type Manuscript
  type Variant
  CopiedBy(Manuscript,Manuscript)
  VariantIn(Manuscript): Variant
  SourceOf(Variant): Manuscript
}

theory T : V {
  ! x : (x ~= SourceOf(VariantIn(x))) =>
        ? y: CopiedBy(y,x) & VariantIn(y) = VariantIn(x).
}
```

Fig. 3 Description of the constraints on the stemmatology problem. This description does not include the actual data mining task, it only provides context for this task.

by which other manuscripts - that is, a hypothesized stemma), and a *partial* specification of VariantIn (for some manuscripts we know which variant they contain). The task is to complete this partial specification in a way that is consistent with the theory; in other words: find an assignment of variants to all manuscripts such that each variant has only one source. This is essentially a satisfiability problem, and IDP's built-in SAT solver is used to solve it.

This approach was used to check the consistency of stemma/dataset pairs. It turned out to solve the problem faster than the custom-built program, and led to the discovery of bugs in that program. The model used by the declarative approach is easy to write down and easy to understand.

This case study illustrates that there exist data analysis tasks for which no solution is readily available, and an algorithmic solution is difficult to construct, yet a declarative description of the problem can easily be built. Flexible modeling tools with powerful reasoning capabilities (more specifically, a combination of first order logic reasoning, optimization, and constraint solving), such as IDP3, can be useful in such cases.

§6 Declarative statistics

6.1 Problematic use of statistics in data mining

As the need for data analysis grows, more and more non-specialist users make use of statistical methods to analyze data. Advanced tools make it possible for anyone to perform advanced statistical analyses. But while using these tools does not require deep knowledge of statistics, using them *correctly* does.

Statistical textbooks warn against common errors such as incorrect use of the t-test, incorrect interpretations of P-values, repeated testing of many different hypotheses, and so on. But in data analysis, more complex and sometimes very subtle issues arise that textbooks do not explicitly warn against, and mistakes are still common in such situations. Several authors have written articles pointing out incorrect use of statistical methods in machine learning and data mining[13, 12], and how to avoid it. While their suggestions are heeded by most researchers, the lack of an active deeper understanding of statistics causes researchers to avoid the mistakes explicitly mentioned in the literature, yet commit comparable mistakes in other situations.

For instance, in cross-validation experiments, standard deviations are often included in tables with results, and statistical significance tests are performed. But it was shown already in 2004 that there is no unbiased estimate of the variance of the cross-validation error estimator.[5] This raises the question what the entities reported as "standard deviations" actually are. Multiple interpretations are possible: they may be the standard deviation of the n estimates obtained in an n-fold cross validation, the standard deviation of the sample proportion in a randomly drawn sample of the same size as the dataset, . . . It is usually not specified in papers exactly what "standard deviation" is being reported; but whatever it is, it cannot be the standard deviation of the accuracy estimate. Yet, it is often interpreted as such.

6.2 Declarative languages for statistical inference

The above examples show that statistical inference is too complex and too subtle to be left to occasional users. Still, the current state of the art is such that correct conclusions can only be drawn if users select the right statistical test (if one exists), perform it correctly, and interpret the results correctly. A possible solution to this problem lies in the use of declarative languages. Ideally, the user should be able to formulate a question (a hypothesis or estimation) and get the answer to that question without having to specify the methodology himself.

Below are some examples of how such a system could work; these are inspired by ongoing research on the topic.[31]

Consider the following problem. We consider three variables describing students: Length, Sex, and StudyProgram. Assume that it is known that both

Length and StudyProgram are dependent on Sex, but both are independent when conditioned on Sex. Assume that we want to estimate the average length of students in the engineering program. Using an SQL-like language for statistical inference, this could look like this:

```
ESTIMATE mean(Length)
FROM Student
WHERE Sex='male' and StudyProgram = 'Engineering'
WITH CONFIDENCE 0.95
```

This query results in a tuple containing an upper and lower bound of the confidence interval for the mean. The width of the obtained confidence interval will depend on the number of students. If not many students are registered in the engineering program, a wide interval will be obtained.

However, if the database contains more students than just those in engineering, the additional information about those students can help obtain a more accurate estimate. Since Length is independent of StudyProgram when conditioned on Sex, the condition `StudyProgram='Engineering'` can simply be dropped. A narrower interval will then be obtained, because more data are used for the estimate.

Now consider the following query, which asks for a point estimate of the mean Length of all students in Engineering:

```
ESTIMATE mean(Length)
FROM Student
WHERE StudyProgram = 'Engineering'
```

If Length were independent of StudyProgram, the condition `StudyProgram='Engineering'` could be dropped, and the full set of students could be used to estimate the mean length of the student population. However, Length is not independent of StudyProgram: some programs are more popular among male students, and male students are on average taller than female students. A correct way of inferring the mean length of all students in engineering would be to estimate the mean length of male and female students separately, using the whole population of students, and then taking a weighted average of these two, using as weights the proportions of male and female students in engineering. A statistician using the statistical language R might for instance write:

```
m  = mean(Length[Sex=='male'])
f = mean(Length[Sex=='female'])
p = length(Length[Sex=='male'])/length(Length)
estimate = p*m+(1-p)*f
```

Note that `mean(Length[Sex=='male'])` refers to the sample mean, not the mean of the population. Its use is correct here because the sample mean is an unbiased estimate of the population mean. A data analyst writing this must be aware of that.

This type of reasoning is pretty standard in probabilistic graphical models. But it is not obvious that a typical researcher who wants to analyze some data knows about such methods, and even if he does, performing the computations requires a significant effort. With the current, procedural, approach to statistics, the researcher would have to think of this option, know how to apply it correctly, and enter the correct formulas for the computation manually. In a declarative setting, the above query would simply give the correct answer; the user does not need to know how it was computed.

So, depending on the execution strategy, the above query could give two different estimates: the sample mean of all engineering students, or an estimate based on a linear combination of the sample means of all male and female students, not restricted to engineering. The system should choose the most accurate among these.

Since the above query asks for a point estimate, it is not clear to the user just how accurate that estimate is. The user could ask for a confidence interval:

```
ESTIMATE mean(Length)
FROM Student
WHERE StudyProgram = 'Engineering'
WITH CONFIDENCE 0.95
```

With the first execution strategy, the computation of this confidence interval must take into account the deviation between the sample mean and the population mean. This is pretty standard. With the second execution strategy, the computation must take into account the deviation between the sample mean and the population mean of Length among male students, the same among female students, and the deviation between the sample proportion of male students and the population proportion. That is, the variance of the weighted average estimator has to be computed. This computation is pretty standard for a statistician, but not trivial for users who do not master the fundamentals of probability theory.

The above case study shows how much intelligence about inductive inference (including the efficiency of alternative ways of estimating parameters) could in principle be built into the system, freeing the user from concerns about correctness or efficiency.

It is noteworthy that the above example makes use of background knowledge about independence of variables, knowledge that would typically be available in the structure of a probabilistic graphical model. Without that knowledge, the above techniques could not be used. The knowledge could be obtained by analyzing the data, but as the result of such an analysis is not certainly correct, uncertainty about it should be taken into account. To the author's knowledge, it is currently not known how this can be done.

The above is just an illustration of how declarative languages for statistical inference could work, and how useful they could be. It is clear that more research is needed before such approaches will be widely useful in practice, but the potential impact of such research is large.

§7 Conclusions

The importance and visibility of Data Science has increased rapidly in the last few decades, and will continue to increase. Ever more "amateur" data scientists are now analyzing data, sometimes using off-the-shelf tools, sometimes by writing programs on their own. Off-the-shelf tools often do not offer the flexibility that is needed, and writing one's own tools is labor-intensive and error-prone. Declarative data analysis has the potential to offer a middle road that combines flexibility with ease of use, efficiency, and correctness guarantees.

Research on declarative data analysis is still ongoing. Much remains to be done, but it has a high potential to shape the future of data science. This article has identified a number of highly relevant research areas, including inductive databases and query languages, constraint-based data mining, declarative modeling languages, and declarative languages for statistical inference. It has presented a selection of past and current research in these areas, as well as opportunities for further research on declarative data analysis. Hopefully, it may inspire researchers to contribute to this emerging and very exciting field.

Acknowledgment Part of this research was supported by the Research Foundation – Flanders (FWO, project G.0682.11) and the Research Council of KU Leuven (GOA/13/010). The author thanks Masayuki Numao and Luc De Raedt for some very useful comments on an earlier draft of this article.

References

1) Antoine Adam, Hendrik Blockeel, Sander Govers, and Abram Aertsen. SCCQL: A constraint-based clustering system. In *Lecture Notes in Computer Sci-*

ence, European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD), Prague, 23-27 September 2013, pages 681–684. Springer, September 2013.

2) Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

3) Tara Andrews, Hendrik Blockeel, Bart Bogaerts, Maurice Bruynooghe, Marc Denecker, Stef De Pooter, Caroline Macé, and Jan Ramon. Analyzing manuscript traditions using constraint-based data mining. In *Proceedings First Workshop on Combining Constraint Solving with Mining and Learning (ECAI 2012 Workshop), First Workshop on Combining Constraint Solving with Mining and Learning, Montpellier, France, 27 August 2012*, pages 15–20, August 2012.

4) Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6:937–965, 2005.

5) Yoshua Bengio and Yves Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, 5:1089–1105, 2004.

6) Hendrik Blockeel, Toon Calders, Élisa Fromont, Bart Goethals, Adriana Prado, and Céline Robardet. An inductive database system based on virtual mining views. *Data Min. Knowl. Discov.*, 24(1):247–287, 2012.

7) Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

8) Maurice Bruynooghe, Hendrik Blockeel, Bart Bogaerts, Broes De Cat, Stef De Pooter, Joachim Jansen, Anthony Labarre, Jan Ramon, Marc Denecker, and Sicco Verwer. Predicate logic as a modeling language: Modeling and solving some machine learning and data mining problems with IDP3. *Theory and Practice of Logic Programming*, 2014. Accepted.

9) Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. A declarative framework for constrained clustering. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III*, pages 419–434, 2013.

10) Ian Davidson. Clustering with constraints. In *Encyclopedia of Database Systems*, pages 393–396. Springer, 2009.

11) Stef De Pooter, Johan Wittocx, and Marc Denecker. A prototype of a knowledge-based programming environment. In *Proceedings of the 19th International Conference on Applications of Declarative Programming and Knowledge Management (INAP 2011), International Conference on Applications of Declarative Programming and Knowledge Management (INAP), Vienna, 28-30 September 2011*, page 6, August 2011.

12) Janez Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

13) Thomas G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

14) Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In Raghu Ramakrishnan, Salvatore J. Stolfo, Roberto J. Bayardo, and Ismail Parsa, editors, *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000*, pages 71–80. ACM, 2000.

15) S. Džeroski, B. Goethals, and P. Panov, editors. *Inductive Databases and Constraint-Based Data Mining.* Springer, 2010.

16) William J. Frawley, Gregory Piatetsky-shapiro, and Christopher J. Matheus. Knowledge discovery in databases: an overview. *AI Magazine*, 13, 1992.

17) Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. Mining sequential patterns with regular expression constraints. *IEEE Trans. Knowl. Data Eng.*, 14(3):530–552, 2002.

18) Alex Guazzelli, Michael Zeller, Wen-Ching Lin, and Graham Williams. PMML: An open standard for sharing models. *The R Journal*, 1(1):60–65, 2009.

19) Tias Guns, Anton Dries, Guido Tack, Siegfried Nijssen, and Luc De Raedt. Miningzinc: A modeling language for constraint-based mining. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, 2013.

20) Pan Hu, Celine Vens, Bart Verstrynge, and Hendrik Blockeel. Generalizing from example clusters. In *Lecture Notes in Computer Science, Discovery Science, Singapore, 6-9 October 2013*, pages 64–78. Springer, October 2013.

21) Tomasz Imielinski and Heikki Mannila. A database perspective on knowledge discovery. *Commun. ACM*, 39(11):58–64, 1996.

22) Rosa Meo, Giuseppe Psaila, and Stefano Ceri. A new sql-like operator for mining association rules. In *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 122–133. Morgan Kaufmann, 1996.

23) Stephen Muggleton. Inductive logic programming. *New Generation Comput.*, 8(4):295–318, 1991.

24) Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *J. Log. Program.*, 19/20:629–679, 1994.

25) Siegfried Nijssen and Tias Guns. Integrating constraint programming and itemset mining. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part II*, pages 467–482, 2010.

26) J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

27) J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

28) Luc De Raedt. *Logical and relational learning.* Cognitive Technologies. Springer, 2008.

29) Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, pages 1453–1484, 2005.

30) Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, pages 667–685. Springer, 2010.

31) Gitte Vanwinckelen and Hendrik Blockeel. A declarative query language for statistical inference. ECML/PKDD 2013 Workshop: Languages for Data Mining and Machine Learning, Prague, Czech Republic, 23 September 2013, September 2013.

32) Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 577–584. Morgan Kaufmann, 2001.

33) Yuyi Wang, Jan Ramon, and Thomas Fannes. An efficiently computable subgraph pattern support measure: counting independent observations. *Data Min. Knowl. Discov.*, 27(3):444–477, 2013.

34) Weifeng Zhi, Xiang Wang, Buyue Qian, Patrick Butler, Naren Ramakrishnan, and Ian Davidson. Clustering with complex constraints - algorithms and applications. In Marie desJardins and Michael L. Littman, editors, *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA.* AAAI Press, 2013.