The Many Roles of Constraints in Data Mining

Letter from the Guest Editor

Roberto J. Bayardo Jr.
IBM Almaden Research Center
650 Harry Rd.
San Jose, CA 95120
bayardo@alum.mit.edu

Effective data mining almost always involves addressing and accounting for multiple trade-offs. Common examples include:

- precision vs. recall
- support vs. confidence
- concise and understandable models versus highly accurate "black boxes"
- sample size vs. error rate
- pattern language or model expressiveness vs. compute time

Constraints play a critical role in data mining precisely because of these numerous trade-offs. Because our mining algorithms are rarely able to produce optimal results with respect to all that are relevant, constraints allow the data miner to focus the algorithm on regions of the trade-off curves (or space) known (or believed) to be most promising. In effect, the ability to express or exploit constraints allows the data miner to inject knowledge into the data mining process.

For this issue of SIGKDD Explorations, I specifically solicited submissions that explore the role of constraints in data mining and knowledge discovery. The result you see here contains eight papers covering an interesting and diverse range of topics.

In the first paper entitled "Classification Trees for Problems with Monotonicity Constraints," Potharst and Feelders examine how to mine classification trees where the predicted class variable is a monotone function of its predictors. Monotonicity is a common requirement in many applications. For example, as Potharst and Feelders note, a selection procedure for a job applicant should be monotone in the performance of the applicant on any test given. Put more simply, a higher test score should never decrease one's probability of getting the job. Surprisingly, even when training data completely satisfies the monotonicity property, algorithms such as CART or C4.5 generally do not produce monotone trees. Potharst and Feelders survey methods that have been proposed for generating trees that do satisfy the *monotonicity constraint*, both when the training data is monotone and when it is not.

Babu, Garofalakis, and Rastogi explore a novel lossy data-table compression method in their paper entitled "SPARTAN: Using Constrained Models for Guaranteed-Error Semantic Compression." SPARTAN works by selecting a subset of attributes (columns) that are not to be explicitly stored. Instead, SPARTAN stores concise classification and regression trees to predict the value of those attributes. What makes this method pertinent to this issue is in how it allows its users to specify hard *error tolerance constraints* on each attribute. More specifically, for numeric attributes, one can specify that the deviation of any predicted value from its actual value in the original data-set is no more than the error-tolerance constraint parameter. For categorical attributes, one can specify an error tolerance that defines an upper-bound on the probability that the approximate value is different from the actual value.

The next paper in this issue, "Learning Missing Values from Summary Constraints" by Wu and Barbará, studies a problem that can be thought of as the inverse of data-table compression. The question addressed in this work is how missing values in a data-set can be inferred from summary data such as a datacube. Inferring the missing values amounts to satisfying the constraints effectively imposed by the summary data. Wu and Barbará examine methods for missing value reconstruction that include techniques from linear algebra, entropy theory, and constraint programming.

The remaining papers in this issue address pushing (or not pushing!) constraints into frequent itemset mining. Pei and Han, in "Constraint Frequent Pattern Mining: a Pattern-Growth View", survey multiple classes of constraints that can be effectively pushed into pattern-growth (and other dynamic tree search) algorithms for frequent itemsets. Moreover, they demonstrate that the pattern-growth method allows a wider class of constraints to be easily pushed into sequential pattern mining.

Leung, Lakshmanan, and Ng focus on how to push a class of constraints termed *succinct* into the pattern growth method. A succinct constraint is one in which there exists a generating function that can enumerate all and only those itemsets that satisfy the constraint. Succinct constraints, if exploited properly, avoid the often costly process of generating and then testing itemsets that cannot possibly satisfy the constraints. The algorithms proposed in this paper widen the class of succinct constraints that can be exploited in this manner.

A contrarian though well motivated view to constraint pushing is presented by Hipp and Güntzer in their position paper "Is Pushing Constraints Deeply into the Mining Algorithms Really What We Want?" Here they argue that in over zealously pushing constraints into itemset mining, we risk reducing the mining process to one of hypothesis testing instead of discovery. They then propose that an initial mining run should instead attempt to identify all possible results of interest, allowing the result to be refined in an iterative process. Such refinements of a general mining result, they claim, is often much more efficient than the repeated mining runs that would be required when mining with constraints. Hipp and Güntzer do point out that some constraints cannot be straightforwardly enforced through general mining result refinement. One such class of constraints they term *row-restriction constraints*, since these constraints specify a subset of data-set rows to which a mining query should be restricted. However, they also show how techniques based on derived items representing row-subset membership can be used to overcome this difficulty.

In the final paper on the constraints theme, "Discovery in Multi-Attribute Data with User-defined Constraints," Perng, Wang, Ma, and Hellerstein consider a problem whose search space is so large that one has no choice but to enforce user-defined constraints during mining. The problem is mining frequent itemsets in relational data with multiple attributes, where the set of attributes used for grouping values in to "transactions" is not fixed in advance. They present monotonicity properties of this itemset mining variant that allow algorithms considerably more efficient than simply applying apriori-like algorithms on each choice of attributes for defining transactions and items.

Outside the constraints theme of this issue of SIGKDD Explorations, the final paper should be a valuable read for anyone interested in data clustering. Aptly entitled "Why so many clustering algorithms – A Position Paper", this report by Estivill-Castro argues that there are many clustering algorithms because the notion of cluster cannot be precisely defined. Estivill-Castro provides a detailed and illuminating analysis of this relatively simple point, culminating in a list of recommendations and maxims one should consider when applying clustering in the knowledge discovery process.

About the Guest Editor

Roberto Bayardo received his BS and MS degrees in Computer Science and Engineering from the Massachusetts Institute of Technology in 1991. He earned a Ph.D. in Computer Sciences from the University of Texas at Austin in 1997. Since graduating, Roberto has worked at the IBM Almaden Research Center in the Quest project. In addition to data-mining and knowledge discovery, Roberto's research interests include databases, web services, peer-to-peer systems, constraint processing, and algorithms for propositional satisfiability.