Graph-Based Data Mining

Lawrence B. Holder
University of Texas at Arlington, USA

Diane J. Cook
University of Texas at Arlington, USA

INTRODUCTION

Graph-based data mining represents a collection of techniques for mining the relational aspects of data represented as a graph. Two major approaches to graph-based data mining are *frequent subgraph mining* and *graph-based relational learning*. This article will focus on one particular approach embodied in the Subdue system, along with recent advances in graph-based supervised learning, graph-based hierarchical conceptual clustering, and graph-grammar induction.

Most approaches to data mining look for associations among an entity's attributes, but relationships between entities represent a rich source of information, and ultimately knowledge. The field of *multi-relational data mining*, of which graph-based data mining is a part, is a new area investigating approaches to mining this relational information by finding associations involving multiple tables in a relational database. Two main approaches have been developed for mining relational information: logic-based approaches and graph-based approaches.

Logic-based approaches fall under the area of *inductive logic programming* (ILP). ILP embodies a number of techniques for inducing a logical theory to describe the data, and many techniques have been adapted to multi-relational data mining (Dzeroski & Lavrac, 2001; Dzeroski, 2003). Graph-based approaches differ from logic-based approaches to relational mining in several ways, the most obvious of which is the underlying representation. Furthermore, logic-based approaches rely on the prior identification of the predicate or predicates to be mined, while graph-based approaches are more data-driven, identifying any portion of the graph that has high support. However, logic-based approaches allow the expression of more complicated patterns involving, for example, recursion, variables, and constraints among variables. These representational limitations of graphs can be overcome, but at a computational cost.

BACKGROUND

Graph-based data mining (GDM) is the task of finding novel, useful, and understandable graph-theoretic patterns in a graph representation of data. Several approaches to GDM exist based on the task of identifying frequently occurring subgraphs in graph transactions, that is, those subgraphs meeting a minimum level of support. Washio & Motoda (2003) provide an excellent survey of these approaches. We here describe four representative GDM methods.

Kuramochi and Karypis (2001) developed the FSG system for finding all frequent subgraphs in large graph databases. FSG starts by finding all frequent single and double edge subgraphs. Then, in each iteration, it generates candidate subgraphs by expanding the subgraphs found in the previous iteration by one edge. In each iteration the algorithm checks how many times the candidate subgraph occurs within an entire graph. The candidates, whose frequency is below a user-defined level, are pruned. The algorithm returns all subgraphs occurring more frequently than the given level.

Yan and Han (2002) introduced gSpan, which combines depth-first search and lexicographic ordering to find frequent subgraphs. Their algorithm starts from all frequent one-edge graphs. The labels on these edges together with labels on incident vertices define a code for every such graph. Expansion of these one-edge graphs maps them to longer codes. Since every graph can map to many codes, all but the smallest code are pruned. Code ordering and pruning reduces the cost of matching frequent subgraphs in gSpan. Yan & Han (2003) describe a refinement to gSpan, called CloseGraph, which identifies only subgraphs satisfying the minimum support, such that no supergraph exists with the same level of support.

Inokuchi et al. (2003) developed the Apriori-based Graph Mining (AGM) system, which searches the space of frequent subgraphs in a bottom-up fashion, beginning

with a single vertex, and then continually expanding by a single vertex and one or more edges. AGM also employs a canonical coding of graphs in order to support fast subgraph matching. AGM returns association rules satisfying user-specified levels of support and confidence.

The last approach to GDM, and the one discussed in the remainder of this chapter, is embodied in the Subdue system (Cook & Holder, 2000). Unlike the above systems, Subdue seeks a subgraph pattern that not only occurs frequently in the input graph, but also significantly compresses the input graph when each instance of the pattern is replaced by a single vertex. Subdue performs a greedy search through the space of subgraphs, beginning with a single vertex and expanding by one edge. Subdue returns the pattern that maximally compresses the input graph. Holder & Cook (2003) describe current and future directions in this graph-based relational learning variant of GDM.

MAIN THRUST

As a representative of GDM methods, this section will focus on the Subdue graph-based data mining system. The input data is a directed graph with labels on vertices and edges. Subdue searches for a substructure that best compresses the input graph. A *substructure* consists of a subgraph definition and all its occurrences throughout the graph. The initial state of the search is the set of substructures consisting of all uniquely labeled vertices. The only operator of the search is the *Extend Substructure* operator. As its name suggests, it extends a substructure in all possible ways by a single edge and a vertex, or by only a single edge if both vertices are already in the subgraph.

Subdue's search is guided by the *minimum description length* (MDL) principle, which seeks to minimize the description length of the entire data set. The evaluation heuristic based on the MDL principle assumes that the best substructure is the one that minimizes the description length of the input graph when compressed by the substructure. The description length of the substructure S given the input graph G is calculated as $DL(G,S) = DL(S)+DL(G|S)$, where $DL(S)$ is the description length of the substructure, and $DL(G|S)$ is the description length of the input graph compressed by the substructure. Subdue seeks a substructure S that minimizes $DL(G,S)$.

The search progresses by applying the *Extend Substructure* operator to each substructure in the current state. The resulting state, however, does not contain all the substructures generated by the *Extend Substructure* operator. The substructures are kept on a queue and are ordered based on their description length (or sometimes referred to as *value*) as calculated using the MDL principle. The queue's length is bounded by a user-defined constant.

The search terminates upon reaching a user-specified limit on the number of substructures extended, or upon exhaustion of the search space. Once the search terminates and Subdue returns the list of best substructures found, the graph can be compressed using the best substructure. The compression procedure replaces all instances of the substructure in the input graph by single vertices, which represent the substructure's instances. Incoming and outgoing edges to and from the replaced instances will point to, or originate from the new vertex that represents the instance. The Subdue algorithm can be invoked again on this compressed graph.

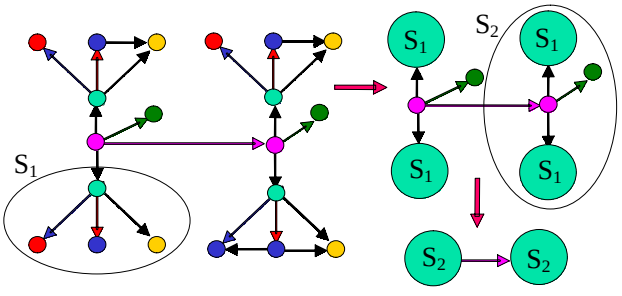
Figure 1 illustrates the GDM process on a simple example. Subdue discovers substructure S_1, which is used to compress the data. Subdue can then run for a second iteration on the compressed graph, discovering substructure S_2. Because instances of a substructure can appear in slightly different forms throughout the data, an inexact graph match, based on graph edit distance, is used to identify substructure instances.

Most GDM methods follow a similar process. Variations involve different heuristics (e.g., frequency vs. MDL) and different search operators (e.g., merge vs. extend).

Graph-Based Hierarchical Conceptual Clustering

Given the ability to find a prevalent subgraph pattern in a larger graph and then compress the graph with this pattern, iterating over this process until the graph can no longer be compressed will produce a hierarchical, conceptual clustering of the input data. On the i^{th} iteration, the best subgraph S_i is used to compress the input graph, introducing new vertices labeled S_i in the graph input to the next iteration. Therefore, any subsequently-discovered subgraph S_j can be defined in terms of one or more of S_is, where $i < j$. The result is a *lattice*, where each cluster can be defined in terms of more than one parent

Figure 1. Graph-based data mining: A simple example

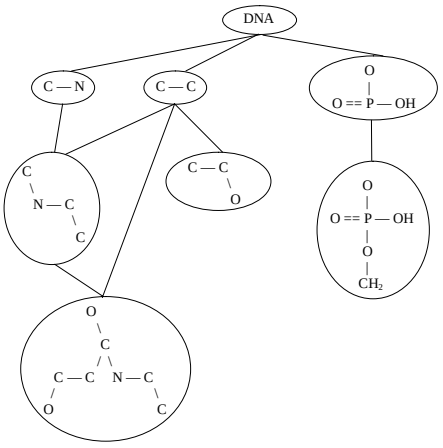


subgraph. For example, shows such a clustering done on a DNA molecule. Note that the ordering of pattern discovery can affect the parents of a pattern. For instance, the lower-left pattern in could have used the C-C-O pattern, rather than the C-C pattern, but in fact, the lower-left pattern is discovered before the C-C-O pattern. For more information on graph-based clustering, see Jonyer et al. (2001).

Graph-Based Supervised Learning

Extending a graph-based data mining approach to perform supervised learning involves the need to handle negative examples (focusing on the two-class scenario). In the case of a graph the negative information can come in three forms. First, the data may be in the form of numerous smaller graphs, or graph transactions, each labeled either positive or negative. Second, data may be composed of two large graphs: one positive and one negative. Third, the data may be one large graph in which the positive and negative labeling occurs throughout. We will talk about the third scenario in the section on future directions. The first scenario is closest to the standard supervised learning problem in that we have a set of clearly defined examples. Let G^+ represent the set of positive graphs, and G^- represent the set of negative graphs. Then, one approach to supervised learning is to find a subgraph that appears often in the positive graphs, but not in the negative graphs. This amounts to replacing the information-theoretic measure with simply an *error-based measure*. This approach will lead the search toward a small subgraph that discriminates well. However, such a subgraph does not necessarily compress well, nor represent a characteristic description of the target concept.

Figure 2. Graph-based hierarchical, conceptual clustering of a DNA molecule



We can bias the search toward a more characteristic description by using the information-theoretic measure to look for a subgraph that compresses the positive examples, but not the negative examples. If $I(G)$ represents the description length (in bits) of the graph G, and $I(G|S)$ represents the description length of graph G compressed by subgraph S, then we can look for an S that minimizes $I(G^+|S) + I(S) + I(G^-) - I(G^-|S)$, where the last two terms represent the portion of the negative graph incorrectly compressed by the subgraph. This approach will lead the search toward a larger subgraph that characterizes the positive examples, but not the negative examples.

Finally, this process can be iterated in a set-covering approach to learn a disjunctive hypothesis. If using the error measure, then any positive example containing the learned subgraph would be removed from subsequent iterations. If using the information-theoretic measure, then instances of the learned subgraph in both the positive and negative examples (even multiple instances per example) are compressed to a single vertex. Note that the compression is a lossy one, that is we do not keep enough information in the compressed graph to know how the instance was connected to the rest of the graph. This approach is consistent with our goal of learning general patterns, rather than mere compression. For more information on graph-based supervised learning, see Gonzalez et al. (2002).

Graph Grammar Induction

As mentioned earlier, two of the advantages of logic-based approach to relational learning are the ability to learn recursive hypotheses and constraints among variables. However, there has been much work in the area of graph grammars, which overcome this limitation. Graph grammars are similar to string grammars except that terminals can be arbitrary graphs rather than symbols from an alphabet. While much of the work on graph grammars involves the analysis of various classes of graph grammars, recent research has begun to develop techniques for learning graph grammars (Doshi et al., 2002; Jonyer et al., 2002).

Figure 3b shows an example of a *recursive graph grammar* production rule learned from the graph in *a*. A GDM approach can be extended to consider graph grammar productions by analyzing the instances of a subgraph to see how they are related to each other. If two or more instances are connected to each other by one or more edges, then a recursive production rule generating an infinite sequence of such connected subgraphs can be constructed. A slight modification to the information-theoretic measure taking into account the extra information needed to describe the recursive

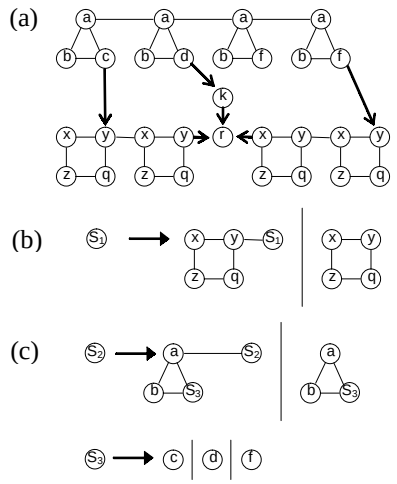
component of the production is all that is needed to allow such a hypothesis to compete along side simple subgraphs (i.e., terminal productions) for maximizing compression.

These graph grammar productions can include non-terminals on the right-hand side. These productions can be disjunctive, as in *c*, which represents the final production learned from *a* using this approach. The disjunction rule is learned by looking for similar, but not identical, extensions to the instances of a subgraph. A new rule can be constructed that captures the disjunctive nature of this extension, and included in the pool of production rules competing based on their ability to compress the input graph. With a proper encoding of this disjunction information, the MDL criterion will tradeoff the complexity of the rule with the amount of compression it affords in the input graph. An alternative to defining these disjunction non-terminals is to instead construct a variable whose range consists of the different disjunctive values of the production. In this way we can introduce constraints among variables contained in a subgraph by adding a constraint edge to the subgraph. For example, if the four instances of the triangle structure in *a* each had another edge to a *c*, *d*, *f* and *f* vertex respectively, then we could propose a new subgraph, where these two vertices are

represented by variables, and an equality constraint is introduced between them. If the range of the variable is numeric, then we can also consider inequality constraints between variables and other vertices or variables in the subgraph pattern.

Jonyer (2003) has developed a graph grammar learning approach with the above capabilities. The approach has shown promise both in handling noise and learning recursive hypotheses in many different domains including learning the building blocks of proteins and communication chains in organized crime.

Figure 3. Graph grammar learning example with (a) the input graph, (b) the first grammar rule learned, and (c) the second and third grammar rules learned



FUTURE TRENDS

The field of graph-based relational learning is still young, but the need for practical algorithms is growing fast. Therefore, we need to address several challenging scalability issues, including incremental learning in dynamic graphs. Another issue regarding practical applications involves the blurring of positive and negative examples in a supervised learning task, that is, the graph has many positive and negative parts, not easily separated, and with varying degrees of class membership.

Partitioning and Incremental Mining for Scalability

Scaling GDM approaches to very large graphs, graphs too big to fit in main memory, is an ever-growing challenge. Two approaches to address this challenge are being investigated. One approach involves *partitioning* the graph into smaller graphs that can be processed in a distributed fashion (Cook et al., 2001). A second approach involves implementing GDM within a relational database management system, taking advantage of user-defined functions and the optimized storage capabilities of the RDBMS.

A newer issue regarding scalability involves *dynamic graphs*. With the advent of real-time streaming data, many data mining systems must mine incrementally, rather than off-line from scratch. Many of the domains we wish to mine in graph form are dynamic domains. We do not have the time to periodically rebuild graphs of all the data to date and run a GDM system from scratch. We must develop methods to incrementally update the graph and the patterns currently prevalent in the graph. One approach is similar to the graph partitioning approach for distributed processing. New data can be stored in an increasing number of partitions. Information within partitions can be exchanged, or a repartitioning can be performed if the information loss exceeds some threshold. GDM can be used to search the new partitions, suggesting new subgraph patterns as they evaluate highly in new and old partitions.

Supervised Learning with Blurred Graphs

In a highly relational domain the positive and negative examples of a concept are not easily separated. Such a

graph is called a *blurred graph*, in that the graph as a whole contains class information, but perhaps not individual components of the graph. This scenario presents a challenge to any data mining system, but especially to a GDM system, where clearly classified data may be tightly related to less classified data. Two approaches to this task are being investigated. The first involves modifying the MDL encoding to take into account the amount of information necessary to describe the class membership of compressed portions of the graph. The second approach involves treating the class membership of a vertex or edge as a cost and weighting the information-theoretic value of the subgraph patterns by the costs of the instances of the pattern. The ability to learn from blurred graphs will allow the user more flexibility in indicating class membership where known, and to varying degrees, without having to clearly separate the graph into disjoint examples.

CONCLUSION

Graph-based data mining (GDM) is a fast-growing field due to the increasing interest in mining the relational aspects of data. We have described several approaches to GDM including logic-based approaches in ILP systems, graph-based frequent subgraph mining approaches in AGM, FSG and gSpan, and a graph-based relational learning approach in Subdue. We described the Subdue approach in detail along with recent advances in supervised learning, clustering, and graph-grammar induction.

However, much work remains to be done. Because many of the graph-theoretic operations inherent in GDM are NP-complete or definitely not in P, scalability is a constant challenge. With the increased need for mining streaming data, the development of new methods for incremental learning from dynamic graphs is important. Also, the blurring of example boundaries in a supervised learning scenario gives rise to graphs, where the class membership of even nearby vertices and edges can vary considerably. We need to develop better methods for learning in these blurred graphs.

Finally, we discussed several domains throughout this paper that benefit from a graphical representation and the use of GDM to extract novel and useful patterns. As more and more domains realize the increased predictive power of patterns in relationships between entities, rather than just attributes of entities, graph-based data mining will become foundational to our ability to better understand the ever-increasing amount of data in our world.

REFERENCES

Cook, D., & Holder, L. (2000). Graph-based data mining. *IEEE Intelligent Systems*, *15*(2), 32-41.

Cook, D., Holder, L., Galal, G., & Maglothin, R. (2001). Approaches to parallel graph-based knowledge discovery. *Journal of Parallel and Distributed Computing*, *61*(3), 427-446.

Doshi, S., Huang, F., & Oates, T. (2002). Inferring the structure of graph grammars from data. In *Proceedings of the International Conference on Knowledge-based Computer Systems*.

Dzeroski, S., & Lavrac, N. (2001). *Relational data mining*. Berlin: Springer Verlag.

Dzeroski, S. (2003). Multi-relational data mining: An introduction. *SIGKDD Explorations*, *5*(1), 1-16.

Gonzalez, J., Holder, L., & Cook, D. (2002). Graph-based relational concept learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*.

Holder, L., & Cook, D. (2003). Graph-based relational learning: Current and future directions. *SIGKDD Explorations*, *5*(1), 90-93.

Inokuchi, A., Washio, T., & Motoda, H. (2003). Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, *50*, 321-254.

Jonyer, I., Cook, D., & Holder, L. (2001). Graph-based hierarchical conceptual clustering. *Journal of Machine Learning Research*, *2*, 19-43.

Jonyer, I., Holder, L., & Cook, D. (2002). Concept formation using graph grammars. In *Proceedings of the KDD Workshop on Multi-Relational Data Mining*.

Jonyer, I. (2003). *Context-free graph grammar induction using the minimum description length principle*. Ph.D. thesis. Department of Computer Science and Engineering, University of Texas at Arlington.

Kuramochi, M., & Karypis, G. (2001). Frequent subgraph discovery. In *Proceedings of the First IEEE Conference on Data Mining*.

Washio, T., & Motoda, H. (2003). State of the art of graph-based data mining. *SIGKDD Explorations*, *5*(1), 59-68.

Yan, X., & Han, J. (2002). Graph-based substructure pattern mining. In *Proceedings of the International Conference on Data Mining*.

Yan, X., & Han, J. (2003). CloseGraph: Mining closed frequent graph patterns. In *Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining*.

KEY TERMS

Blurred Graph: Graph in which each vertex and edge can belong to multiple categories to varying degrees. Such a graph complicates the ability to clearly define transactions on which to perform data mining.

Conceptual Graph: Graph representation described by a precise semantics based on first-order logic.

Dynamic Graph: Graph representing a constantly changing stream of data.

Frequent Subgraph Mining: Finding all subgraphs within a set of graph transactions whose frequency satisfies a user-specified level of minimum support.

Graph-Based Data Mining: Finding novel, useful, and understandable graph-theoretic patterns in a graph representation of data.

Graph Grammar: Grammar describing the construction of a set of graphs, where terminals and nonterminals represent vertices, edges or entire subgraphs.

Inductive Logic Programming: Techniques for learning a first-order logic theory to describe a set of relational data.

Minimum Description Length (MDL) Principle: Principle stating that the best theory describing a set of data is the one minimizing the description length of the theory plus the description length of the data described (or compressed) by the theory.

Multi-Relational Data Mining: Mining patterns that involve multiple tables in a relational database.