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.

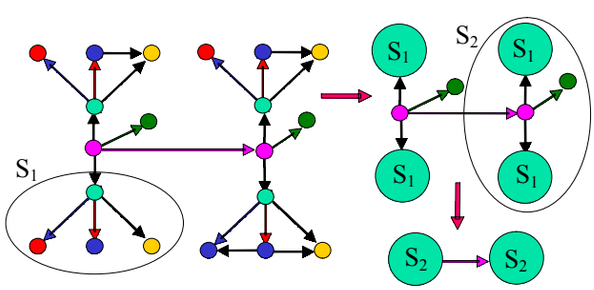
Figure 1 illustrates the GDM process on a simple example. Subdue discovers substructure S_1, which is used to compress the data. Subdue can then run for a second iteration on the compressed graph, discovering substructure S_2. Because instances of a substructure can appear in slightly different forms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



Related Content