Graph Based Concept Learning

Jesus Gonzalez, Lawrence B. Holder, Diane J. Cook
University of Texas at Arlington, Department of Computer Science and Engineering
Box 19015, Arlington, TX 76019-0015
{gonzalez,holder,cook}@cse.uta.edu
URL: http://cygnus.uta.edu/subdue/ConceptLearning

Introduction

Concept Learning is a Machine Learning technique in which the learning process is driven by providing positive and negative examples to the learner. From those examples, the learner builds a hypothesis (concept) that describes the positive examples and excludes the negative examples. Inductive Logic Programming (ILP) systems have successfully been used as concept learners. Examples of those are Foil [1] and Progol [5]. The main engine of these systems is based in first order logic. In this research we introduce a graph based relational concept learning system called SubdueCL, which through the experiments has shown that it is competitive with ILP systems in different types of domains. SubdueCL is an extension made to the Subdue [2] system, which is an unsupervised graph based learner. The extensions made to Subdue to provide the concept learning capability were mainly related to the ability to accept as input positive and negative examples and the criteria used to choose the rules to be added to the hypothesis to be learned.

Subdue Discovery System

Given a labeled graph as input, the Subdue system [2] discovers substructures (sub-graphs) that compress the input graph, according to the minimum description length principle [6], and represent structural concepts in the data. Note that the input graph need not be a connected graph, which will be true during concept learning when multiple examples are included in one input graph.

The main discovery algorithm is a computationally-constrained beam search. The algorithm begins with the substructure matching a single vertex in the graph. Each iteration the algorithm selects the best substructure and incrementally expands the instances of the substructure. The algorithm searches for the best substructure until all possible substructures have been considered or the total amount of computation exceeds a given limit. Evaluation of each substructure is determined by how well the substructure compresses the description length of the input graph. The best substructure found by Subdue can be used to compress the input graph, which can then be input to another iteration of Subdue. After several iterations, Subdue builds a hierarchical description of the input data where later substructures are defined in terms of substructures discovered on previous iterations.

Because instances of a substructure can appear in different forms throughout the data, an inexact graph match is used to identify substructure instances with a bounded amount of variation from the substructure definition. Variation is described in terms of basic transformations such as deletion, insertion, and substitution of vertices, edges and/or labels. The fraction of the size of an instance by which the instance can be different from the pattern definition can be specified with the threshold parameter. For example, when a threshold of 0.2 is specified, Subdue will discover all instances of the pattern that have less than 20% difference from the pattern definition. The inexact graph match is constrained to run in polynomial time. Subdue is capable of handling very large graphs (e.g., millions of vertices and edges) by first partitioning the graph so as to minimize the loss of frequent edges, analyzing each partition in parallel, and collaborating to evaluate each substructure to identify a global best substructure [4]. Subdue has been applied to several domains including image analysis, CAD circuit analysis, Chinese character databases, program source code, chemical reaction chains, protein databases, and artificially-generated databases [3, 7].

SubdueCL

Since SubdueCL is an extension to Subdue, it uses Subdue's core functions to perform graph operations, but the learning process is different since it works as a supervised learner by differentiating positive and negative examples using a set-covering approach. The hypothesis found by SubdueCL consists of a set of disjunctions of conjunctions (substructures), i.e., the concept may contain several rules. SubdueCL forms one of these conjunctions (rules) in each iteration. Positive example graphs covered in a previous iteration are removed from the graph for subsequent iterations.

The way in which SubdueCL decides if the substructures (or rules) will be part of the concept or not is also different from Subdue. SubdueCL uses an evaluation formula to give a value to all the generated substructures. This formula assigns a value to the substructure according to how well they describe the positive examples (or a subset of the positive examples) without describing the negative examples. Then, positive examples covered by the substructure increase the substructure value while negative examples decrease its value. In this formula the positive examples that are not covered and the negative examples covered by the substructure are considered errors because the ideal substructure would be one covering all the positive examples without covering any negative example. Then, the substructure value is calculated as follows:

$$value = 1 - \frac{\#PosEgsNotCovered + \#NegEgsCovered}{\#PosEgs + \#NegEgs}$$

Using this formula, SubdueCL chooses rules that maximize the substructure's value and in this way it minimizes the number of errors made by the substructures used to form the concept.

SubdueCL allows hypotheses where some of the rules may cover negative examples. We are working on a version of SubdueCL that produces only consistent hypotheses; this means that the rules of the hypothesis will only cover positive examples without covering any negative example.

Results

As part of the experiments SubdueCL is being compared with the two ILP systems Foil [1] and Progol [5]. The type of domain that is being used corresponds to attribute-value databases. Four databases have been used for the comparison: Golf, Diabetes, Credit, and Vote. The comparison process consists of a ten fold cross validation (that is 9/10 of the examples are used for the training phase and 1/10 for testing) that produces 10 independent results and a significance test for each domain.

The results show that SubdueCL performed better for the Golf, Credit, and Vote domains and Progol performed better for the Diabetes domain. Foil performed second for the Golf, Credit, and Vote domains and third for the Diabetes domain. Progol performed last in the Golf, Credit, and Vote domains. We also compared SubdueCL with C4.5 and it performed slightly better in non numeric domains (Golf, Credit and Vote).

Conclusion

The graph based concept learning system SubdueCL has been shown to be competitive (even better) that the ILP systems Foil and Progol in attribute-value domains. The goal now is to show how SubdueCL performs with relational domains where ILP systems have produced very good results. Some preliminary results using relational domains show that SubdueCL is able to learn the tic-tac-toe domain and produce a perfect output just as Foil and Progol did.

References

[1] R. M. Cameron-Jones and J. R. Quinlan. Efficient top-down induction of logic programs. SIGART Bulletin, 5(1):33-42, 1994.

[2] D. J. Cook, L. B. Holder. Substructure discovery using minimum description length and background knowledge. Journal of Artificial Intelligence Research, 1:231-255, 1994.

[3] D. J. Cook, L. B. Holder, and S. Djoko. Scalable discovery of informative structural concepts using domain knowledge. IEEE Expert, 11(5):59-68, 1996.

[4] G. Galal, D. J. Cook, and L. B. Holder. Improving the scalability in a scientific discovery system by exploiting parallelism. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pages 171-174, 1997.

[5] S. Muggleton. Inverse entailment and Progol. New Generation Computing, 13:245-286, 1995.

[6] J. Rissanen. Stochastic Complexity in Statistical Inquiry. World Scientific Publishing Company, 1989.

[7] S. Su, D. J. Cook, and L. B. Holder. Applications of knowledge discovery to molecular biology: Identifying structural regularities in proteins. In Proceedings of the Pacific Symposium on Biocomputing, pages 190-201, 1999.