Inductive Logic Programming in Haskell

Marcus Peacock 0919274

11th October 2012

Problem

The area of interest for my project is Inductive Logic Programming (ILP). ILP brings together machine learning and logic programming with the aim of producing algorithms which can inductively learn relational descriptions. I am interested in creating an ILP system in Haskell.

Objectives

My initial objective is to create a simple emperical system. This objective can be broken down in to research on what exactly is required for an ILP system, finding or creating a suitable representation for logic, implementing the required methods and finally creating the search and heuristic system. Empirical systems are much less complex than interactive systems and systems which use searching such as depth-first or breadth-first are simpler than systems with use algorithms such as A*. Therefore, as I expect the majority of the time on my project to be spent on the logic aspect, I have set my objective to create a basic empirical system. If this turns out to take less time than I expected I would look at improving the searching algorithm of the system and attempt to make it more efficient.

Methods & Research

The two pieces of work contributing to most of my understanding of ILP are N. Lavrac, S. Dzeroski [1] and S. Muggleton [2].

My project will be split in to two parts. The initial part of the project will be researching the theory and methods behind an ILP system and deciding how I will implement them. The second part of the project will be my Haskell implementation.

My current understanding is that an ILP system is a system which starts with some background knowledge and a set of positive and negative examples and attempts to find a hypothesis which, along with the background knowledge, entails all the positive examples and none of the negative examples. It is also

important to be able to determine if a hypothesis is correct, preferably a hypothesis is complete and consistent but noise must be accounted for and being easily readable by humans is a positive.

The problem can be viewed as a search problem, the system is searching through all the possible hypothesis and searching for one which fits the necessary criteria. As it is impractical for all the hypothesis to be generated and checked, heuristics and search algorithms are used.

An ILP system can either be interactive or empirical. An empirical system starts with all the background knowledge and examples it will use and attempts to find a hypothesis. An interactive system can start with some or none of the information it will use, allows for information to be added and can deal with working hypothesis.

A search of the hypothesis space an be performed bottom-up or top-down. Generalisation techniques search in a bottom-up manner: starting from training examples and use generalization operators. On the other hand, specialization techniques search the hypothesis space top-down, from the most general to specific concept descriptions using specialization operators. My system will be empirical and my understanding is that specialization techniques are better suited for empirical systems and can deal with noise with suitable heuristics. It will therefore be required that I research and understand specialization techniques in order to implement them correctly.

An ILP system requires a language of logic to operate with. I am yet to decide how I will represent the logic of my system, it may be possible to use pre-existing modules for logic representation or I may decide to write my own system in Haskell. Research on what is available will be required to make this decision.

I will test my system using examples that are used with existing ILP systems and compare my results with them. I will also compare how quickly my system runs in comparison. If suitable it would be interesting to see how the results of different types of ILP system compare and how quickly each type runs in comparison to mine.

As my project is in Haskell, extra time will be required to learn how to effectively use Haskell for my project. I have some experience in Haskell but I chose to do my project in Haskell because I am interested in becoming much more proficient. I would like to explore various possibilities for writing the methods for my system and attempt to make them efficient whilst recognising how human readable they are.

Resources

I expect to be able to use vim for text and coding work, along with git and github for backup purposes and ghc for compiling Haskell.

Milestones & Timetable

By the end of October I expect to have done enough research to be able to make decisions on the representation I will use for my project and following that, be able to start implementing the methods required. I aim to have the logic part of the system to be complete before the start of term two, allowing the rest of the time to be used on implementing suitable search algorithms and testing.

Date	Milestone
11th October 2012	Initial Project Specification
31st October 2012	Research complete
26th November 2012	Progress Report
8th December 2012	Logic representation and methods complete
1st February 2013	Functional system with suitable examples
4th February 2013	Project Presentation
25th April 2013	Final Report

References

[1] N. Lavrac and S. Dzeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, 1991.

[2] S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4), 1991.