**Stephen H. Muggleton and Hiroaki Watanabe (Eds.):
Latest advances in inductive logic programming
World Scientific Publishing, 2014, ISBN: 978-1783265084**

Man Leung Wong¹

Published online: 28 December 2015
© Springer Science+Business Media New York 2015

Inductive logic programming (ILP) investigates the construction of logic programs from training examples and background knowledge, where a logic program is a set of sentences in logical form, expressing facts and rules about some domain. ILP is more powerful than traditional machine learning methods, such as C4.5 decision trees, because it uses an expressive first-order logic framework and readily allows the user to incorporate background knowledge. In first-order logic, sentences are formed by using *constants*, *variables*, *predicates*, *functions*, *quantifiers*, and *connectives* in a well-formed format, where *constants* are the objects in the domain of discourse, *variables* range over the objects in the domain, *predicates* specify the properties of objects or the relations among objects, *functions* map object(s) onto other object(s), *quantifiers* are $\forall$ (Universal) and $\exists$ (Existential) that quantify the variables, and *connectives* include negation, and, or, etc.

ILP has a strong theoretical foundation from logic programming and computational learning theory. There has been significant progress in ILP in the last two decades and this book contains 25 papers associated with the 21st International Conference of Inductive Logic Programming which was held in Cumberland Lodge, an educational charity and a unique conference center in the heart of the Great Park, Windsor, United Kingdom. The book is divided into 7 parts and the first part has 9 papers about applications of ILP in different domains. In Part 2 five research studies that extend ILP to probabilistic logical learning are presented. Part 3 contains a number of papers about the implementations of ILP systems. Research studies about theory, logical learning, constraints, as well as spatial and temporal learning are presented in parts 4–7.

✉ Man Leung Wong
mlwong@ln.edu.hk

¹ Department of Computing and Decision Sciences, Lingnan University, Tuen Mun, Hong Kong

Since the book assumes that the readers are post-graduate students studying ILP or expert ILP researchers, extensive knowledge of different kinds of logics, probabilistic logic learning, and logic programming is required to understand most papers in it. Nevertheless, some application-oriented papers are more accessible. Three representative papers in this category will be discussed in the following paragraphs.

Chapter 3 describes a semantic data mining system called g-SEGS that uses additional background knowledge in the learning process. Semantic data are the information that allows machines to understand the meaning of data. Vavpetič and Lavrač say that the amount of semantic data available is increasing rapidly because of the Semantic Web and the availability of numerous ontologies (such as WordNet and the USA Gene Ontology Project). Semantic data mining is a new subfield of data mining in which semantic data are themselves mined.

g-SEGS takes: (1) background knowledge in the form of ontologies; (2) a set of training examples; and (3) an example-to-ontology map which associates each example with one or more concepts from the given ontologies. The hypothesis language of g-SEGS consists of rules of the form of `class(X) ← Condition`.

For example, the following rule specifies that a customer *X* is classified as a big spender if she is a solicitor and uses prestige banking service.

`class(X) ← occupation(X, solicitor) and bankingService(X, prestige)`

where *solicitor* and *prestige* are respectively terms from the occupation and banking service ontologies. g-SEGS uses a top-down bounded exhaustive search algorithm to enumerate all possible rules by taking one term from each ontology. As the number of rules can be large, g-SEGS removes uninteresting and overlapping rules by considering the weighted accuracy of each rule.

Vavpetič and Lavrač compare g-SEGS and Aleph on two bioinformatics problems. In both cases g-SEGS was better. Moreover, g-SEGS generates the rules approximately 20–30 times faster than Aleph.

Chapter 5 (written by Hiroyuki Nishiyama and Fumio Mizoguchi) presents a system that can recommend cosmetic products to users through smart phones. Firstly the user takes a picture of themselves with the phone's camera (a selfie). Then the system analyses the image of their skin in the photograph and determines if it can be diagnosed by using a support vector machine (SVM). If not, the system requests the user retakes the photo. If the image can be diagnosed, the system employs the rules induced by ILP to diagnose the skin condition. Based on the diagnosis, the system then recommends a cosmetic.

Chapter 24 describes an ILP approach to learn soft constraints for scheduling worker shifts. A number of good and bad rosters are provided as training examples and a set of rules are obtained by using the GKS ILP system. The rules differentiate between good and bad rosters. The conditions in these rules which indicate good rosters are then treated as general conditions for good rosters. They are converted into soft constraints in a constraint logic program, which also contains hand coded hard constraints. Yoshihisa Shiina and Hayato Ohwada then use constraint logic

programming to find good rosters that fulfill all the hard constraints while minimizing the number of violations of the soft constraints.

Genetic Programming and Evolvable Machines readers, if they have the background knowledge or are planning to acquire it, will find this book thought-provoking and valuable. They can use the advanced ILP techniques and concepts it contains in their research on genetic programming (GP). For example, researchers can apply inverse resolution, a standard operation of ILP, as a local search operator of GP. The ILP methods can also be incorporated into genetic improvement (GI) systems to improve software applications and specifications if the applications are written in a logic programming language or use logical formalisms. Chapter 23 (Henderson and Muggleton) presents a method that invents functional abstractions from a set of λ -calculus programs. Perhaps the method can be employed in GP to evolve sub-functions and program modules. Current research in grammar-based GP usually specifies the search space and the domain/background knowledge in context-free grammars (e.g. the grammars used in grammatical evolution, GE) or context-sensitive grammars (e.g. first-order logic or logic grammars). Perhaps representing knowledge in higher-order logic (HOL) would help the evolution of programs. In HOL, variables can range over predicates, objects, sets of objects, sets of sets of objects, and so on. Perhaps the ILP learning methods for HOL can be used to modify the search space dynamically and improve the knowledge (i.e. grammars) represented in HOL.