Lookahead and Discretization in ILP

Hendrik Blockeel and Luc De Raedt

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A
3001 Heverlee
e-mail: \{Hendrik.Blockeel, Luc.DeRaedt\}@cs.kuleuven.ac.be

Abstract. We present and evaluate two methods for improving the performance of ILP systems. One of them is discretization of numerical attributes, based on Fayyad and Irani’s text [9], but adapted and extended in such a way that it can cope with some aspects of discretization that only occur in relational learning problems (when indeterminate literals occur). The second technique is lookahead. It is a well-known problem in ILP that a learner cannot always assess the quality of a refinement without knowing which refinements will be enabled afterwards, i.e. without looking ahead in the refinement lattice. We present a simple method for specifying when lookahead is to be used, and what kind of lookahead is interesting. Both the discretization and lookahead techniques are evaluated experimentally. The results show that both techniques improve the quality of the induced theory, while computational costs are acceptable.

1 Introduction

Propositional learning has been studied much more extensively than inductive logic programming (ILP), and at this moment the former field is better understood than the latter. However, ILP shares many techniques, heuristics etc. with propositional learning, and therefore it can often profit from results obtained for propositional learners. Due to several aspects of ILP that do not occur in propositional learning, it is often necessary to adapt these techniques to the specific ILP context.

In this paper, we discuss two such upgrades of propositional learning results to the ILP context. The first is discretization of continuous attributes. Irani and Fayyad [9] have presented a propositional method that divides a continuous domain into several subsets which can then be used as discrete values. We will briefly discuss the method, show that ILP poses some problems with respect to discretization that do not occur in propositional learning, and propose an adaptation of the method.

The second topic we shall discuss, is the use of lookahead. It is a well-known problem with relational learners that most heuristics (e.g. information gain) have problems with assessing the quality of a refinement of a rule, because a single literal that is added by the refinement step may not cause any gain, but may be very important to make the addition of gainful literals possible later on (because
it introduces new variables). The advantage of adding a literal may only become clear further down the refinement tree. In general, heuristics that work well for propositional learners do not always perform as well for relational learners.

We propose a lookahead technique to alleviate this problem. By allowing the learner to look more than one level ahead in the refinement lattice, it may be able to assess the quality of a refinement more accurately.

Both the discretization and lookahead methods have been implemented in a novel ILP system called TILDE, and we present experimental results confirming the usefulness of both techniques.

This text is organized as follows. In Section 2, we briefly discuss the ILP setting that is used. In Section 3 we discuss discretization, in Section 4 lookahead. Conclusions are presented in Section 5.

2 The Learning Setting

We essentially use the learning from interpretations paradigm for inductive logic programming, introduced by [4], and related to other inductive logic programming settings in [3].

In this paradigm, each example is a Prolog knowledge base (i.e. a set of definite clauses), encoding the specific properties of the example. Furthermore, each example is classified into one of a finite set of possible classes. One may also specify background knowledge $B$ in the form of a Prolog knowledge base.

More formally, the problem specification is:

**Given:** a set of classes $C$, a set of classified examples $E$, and a background theory $B$,

**Find:** a hypothesis $H$ (a set of definite clauses in Prolog), such that for all $e \in E$ : $H \land e \land B \models c$, and $H \land e \land B \not\models c'$ where $c$ is the class of the example $e$ and $c' \in C - \{c\}$.

Our experiments have been done with the ILP system TILDE[1], which represents the induced hypotheses as logical decision trees (these are a first order logic upgrade of the classical decision trees used in propositional concept learning).

**Example 1.** Suppose a number of machines are under revision. Some have to be sent back to the manufacturer, and others can be kept. The aim is to predict whether a machine needs to be sent back.

Given the following set of examples (each example represents one machine):

<table>
<thead>
<tr>
<th>Example 1</th>
<th>Example 2</th>
<th>Example 3</th>
<th>Example 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>class(keep)</td>
<td>class(sendback)</td>
<td>class(sendback)</td>
<td>class(keep)</td>
</tr>
<tr>
<td>worn(gear)</td>
<td>worn(engine)</td>
<td>worn(control_unit)</td>
<td></td>
</tr>
<tr>
<td>worn(chain)</td>
<td>worn(chain)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

and the following background knowledge: