Learning from Highly Structured Data by Decomposition*

René Mac Kinney-Romero and Christophe Giraud-Carrier

Department of Computer Science, University of Bristol
Bristol, BS8 1UB, UK
{romero,cgc}@cs.bristol.ac.uk

Abstract. This paper addresses the problem of learning from highly structured data. Specifically, it describes a procedure, called decomposition, that allows a learner to access automatically the subparts of examples represented as closed terms in a higher-order language. This procedure maintains a clear distinction between the structure of an individual and its properties. A learning system based on decomposition is also presented and several examples of its use are described.

1 Introduction

Machine Learning (ML) deals with the induction of general descriptions (e.g., decision trees, neural networks) from specific instances. That is, given a set of examples of a target concept, a ML system induces a representation of the concept, which explains the examples and accurately extends to previously unseen instances of the concept.

Most ML systems use an attribute-value representation for the examples. Although this simple representation allows the building of efficient learners, it also hinders the ability of such systems to handle directly examples and concepts with complex structure. To overcome this limitation, two approaches have been used. The first one is based on data transformation or pre-processing. Here, a structured (e.g., first-order) representation is mapped into an equivalent attribute-value representation by capturing subparts of structures and n-ary predicates ($n > 1$) as new attributes, which can then be manipulated directly by standard attribute-value learners (e.g., see the LINUS system [7]). The second approach consists of "upgrading" the learner. Here, the learner is designed so as to be able to manipulate structured representations directly (e.g., see the ILP framework for first-order concepts [9]).

From a practitioner's standpoint, there is a clear advantage in the second approach since it allows the user to represent the problem in a "natural" way (i.e., consistent with domain-specific standards or practices), without recourse to a pre-processing phase, which is tedious, prone to loss of information and often costly as it may require expert intervention. In keeping with this approach to make ML techniques more readily available to practitioners, our research

* This work is funded in part by grants from CONACYT and UAM, México

J.M. Żytkow and J. Rauch (Eds.): PKDD'99, LNAI 1704, pp. 436–441, 1999.

focuses on the use of sufficiently expressive representation languages so that highly-structured data can be manipulated transparently by the corresponding learning systems. In particular, a novel, higher-order framework, in which examples are closed terms and concepts are Escher programs, has been developed [5]. This paper describes a learning algorithm for the above higher-order framework based on a decomposition method. Decomposition can be regarded as an extension of flattening [10] to higher-order representations, which maintains a clear distinction between the structure of an individual and its properties.

2 Decomposition

The application of machine learning to real-world problems generally requires the mapping of the user's conceptual view of the data to a representation suitable for use by the learner. In addition to being tedious and often costly, this paradigm is undesirable as it may cause a loss of information. We argue that learners should be taylored to representations, rather than the other way around. In other words, the user should be able to describe the problem as is most natural to him/her and leave the learner to manipulate the representation and induce generalisations.

To this end, we have proposed an "individuals-as-terms" representation [4], where examples are highly-structured closed terms in a higher-order logic language. As an example, consider the mutagenicity problem, which is concerned with identifying nitroaromatic and heteroaromatic mutagenic compounds. In this problem, the examples are molecules together with some of their properties. A relatively natural representation for such compounds, as proposed in [2], consists of closed terms, here tuples, of the following form.

```
type Molecule = (Ind1, IndA, Lumo, {Atom}, {Bond})
```

where `Ind1` and `IndA` are boolean values relating to structural properties, `Lumo` is the energy level of the lowest unoccupied molecular orbital, and {`Atom`} and {`Bond`} capture the structure of the molecule as a graph, i.e., a set of atoms and the bonds between them. Atoms and bonds are defined as follows.

```
type Atom = (Label, Element, AtomType, Charge)
type Bond = ({Label}, BondType)
```

Hence, a sample molecule has the following representation.

$$(\text{False}, \text{False}, -1.034, \{(1, \text{C}, 22, -0.128), (10, \text{H}, 3, 0.132),\\ (11, \text{C}, 29, 0.002), ...\}, \{(\{1, 2\}, 7), (\{1, 7\}, 1), ...\})$$

Standard transformational techniques, such as flattening [10] do not apply to such data. We extend these through a technique, called decomposition, whose task is to extract the structural information of the examples, thus allowing the learning tool to access directly sub-parts of the structures being represented.

Decomposition takes a highly structured term and a maximum depth, and returns the set of all possible variables up to the specified depth. A depth of

zero returns only one variable representing the whole structure. A depth of one returns one variable for each component at the top level (e.g., each element of a tuple or set) and so on. The decomposition set consists of tuples of the form

$$\{\langle \text{name}, \text{type}, \text{value}, \text{structural predicates}\rangle\}$$

so that each variable is qualified by its name, type, value and the structural predicates needed to obtain it. The following illustrates the decomposition technique on the the above sample molecule. Assuming a maximum depth of 1, decomposition will return the following set, where v_0 refers to the top-level term.

$$\begin{aligned}&\{\langle v_1, \text{Ind1}, \text{False}, v_0 == (v_1, v_2, v_3, v_4, v_5)\rangle,\\ &\langle v_2, \text{IndA}, \text{False}, v_0 == (v_1, v_2, v_3, v_4, v_5)\rangle,\\ &\langle v_3, \text{Lumo}, -1.034, v_0 == (v_1, v_2, v_3, v_4, v_5)\rangle,\\ &\langle v_4, \{\text{Atom}\}, \{\ldots\}, v_0 == (v_1, v_2, v_3, v_4, v_5)\rangle,\\ &\langle v_5, \{\text{Bond}\}, \{\ldots\}, v_0 == (v_1, v_2, v_3, v_4, v_5)\rangle\}\end{aligned}$$

With a depth of 2, v_4 and v_5 would be further decomposed. The decomposition of v_4, which has value

$$\{(1, \texttt{C}, 22, -0.128), (10, \texttt{H}, 3, 0.132), (11, \texttt{C}, 29, 0.002), \ldots\}$$

for example, would yield

$$\begin{aligned}&\{\langle v_6, \text{Atom}, (1, \text{C}, 22, -0.128), v_0 == (v_1, v_2, v_3, v_4, v_5) \wedge v_6 \in v_4\rangle,\\ &\langle v_7, \text{Atom}, (10, \text{H}, 3, 0.132), v_0 == (v_1, v_2, v_3, v_4, v_5) \wedge v_7 \in v_4\rangle, \ldots\}\end{aligned}$$

The current implementation of the decomposition algorithm supports only lists, tuples and sets. It is possible, however, to devise a syntax in which the user of the learning system is able to give new types along with the information on how to decompose them.

3 Learning by Decomposition

The approach presented has been implemented in ALFIE, an Algorithm for Learning Functions In Escher. This implementation is based on the framework presented in [3], which uses the Escher language [8] as the representation vehicle for examples and concepts.

ALFIE induces concepts in the form of decision lists, i.e.,

$$\text{if } E_1 \text{ then } t_1 \text{ else if } E_2 \text{ then } t_2 \text{ else } \ldots \text{ if } E_n \text{ then } t_m \text{ else } t_0$$

where each E_i is a boolean expression and the t_j's are class labels. The class t_0 is called the *default* and is generally, although not necessarily, the majority class. The algorithm uses sequential covering to find each E_i. It uses the decomposition set of the first example and from it finds the E_1 with the highest accuracy (measured as the information gain on covering). It then computes the set of

Input: decomposition set D of $\mathcal{E}$ and set of properties $P = \{p : \sigma \to \text{Bool}\}$
Output: set of atomic conditions C

$C = \phi$
For all $t \in D$ such that $value(t) == k$
    $C = C \cup \{name(t) == k\}$
For all $t_1, t_2 \in D$ such that $type(t_1) == type(t_2)$ and $value(t_1) == value(t_2)$
    $C = C \cup \{name(t_1) == name(t_2)\}$
For all $t_1, \ldots, t_n \in D$ that match σ, where $p : \sigma \to \text{Bool} \in P$
    $C = C \cup \{p(t_1, \ldots, t_n)\}$
return C

Fig. 1. Algorithm to Generate Atomic Conditions

examples that are not yet covered, selects the first one and repeats this procedure until all examples are covered.

To create the E_i's, ALFIE builds conjunctions of atomic conditions from the decomposed examples as shown in Figure 1. ALFIE uses information about the values of variables to create all possible equality conditions between a variable's value and a constant, and between two variables' values. In addition, the user may provide a set of boolean functions that can be applied to the elements of the decomposition set (set P in Figure 1). These functions represent properties which may be present in the sub-parts of the structure of the data. They constitute the *background knowledge* of the learner and may be quite complex. For example, a function to test whether a molecule contains less than 5 oxygen atoms would have the following (higher-order) form.

$$(\text{card}(\text{filter}(v, v == (l, e, a, c) \wedge a == \text{O})) < 5)$$

To illustrate the algorithm of Figure 1, consider the following partial decomposition of a molecule (set D).

$\{\langle v_3, \text{Bool}, \text{False}, v_0 == (v_1, v_2, v_3, v_4, v_5)\rangle,$
$\langle v_4, \text{IndA}, -1.387, v_0 == (v_1, v_2, v_3, v_4, v_5)\rangle,$
$\langle v_{53}, \text{Atom}, \text{H}, v_0 == (v_1, v_2, v_3, v_4, v_5) \wedge v_{51} \in v_4 \wedge v_{51} == (v_{52}, v_{53}, v_{54}, v_{55})\rangle,$
$\langle v_{98}, \text{Atom}, \text{H}, v_0 == (v_1, v_2, v_3, v_4, v_5) \wedge v_{96} \in v_4 \wedge v_{96} == (v_{97}, v_{98}, v_{99}, v_{100})\rangle,$
$\langle v_{238}, \text{Bond}, \{3, 4\}, v_0 == (v_1, v_2, v_3, v_4, v_5) \wedge v_{237} \in v_5 \wedge v_{237} == (v_{238}, v_{239})\rangle\}$

Assume the following function is also given as background knowledge (set P).

$$(> -2.368) : \text{IndA} \to \text{Bool}$$

Then the set C produced by the algorithm is:

$\{v_3 == \text{False}, v_4 == -1.387, v_{53} == \text{H}, v_{98} == \text{H}, v_{238} == \{3, 4\},$
$v_{53} == v_{98}, v_4 > -2.368\}$

4 Experiments

Although they do not require decomposition, a number of experiments were carried out with attribute-value problems to check ALFIE's ability to generate accurate theories. The results obtained on these problems compare favourably with those of standard learners, such as C4.5. The experiments detailled here focus on learning from highly structured data.

4.1 Bongard

The 47th pattern recognition problem from [1] aims at inducing the concept "there is a circle inside a triangle" from examples consisting of a set of shapes that may contain other shapes inside them. One of the simplest representations consists of encoding a figure as a pair, where the first element is the shape of the figure and the second a set of the figures contained in it.

```
Figure = Null | (Shape, {Figure})
Shape = Circle | Triangle
```

The examples are sets of such figures together with a truth value. Given such a representation, ALFIE produces the solution

$$f(v_1) = \text{if}(v_5 \in v_1 \wedge v_5 == (v_6, v_7) \wedge v_8 \in v_7 \wedge v_8 == (v_9, v_{10}) \\ \wedge v_9 == \text{Circle} \wedge v_6 == \text{Triangle}) \text{ then True else False}$$

whose English equivalent is "if there is a Circle inside a Triangle then True else False."

4.2 Mutagenicity

Mutagenicity is a well known dataset in the ILP community and a number of experiments with this problem are reported in [6]. The dataset consists of 230 chemical compounds, 42 of which have proven difficult to classify automatically. The experiment was carried out on the remaining 188. A chemical compound is represented as a highly-structured closed term consisting of the atoms and bonds as discussed above. An atom has a label, an element, a type (there are 233 such types) and a partial charge (which is a real number). A bond connects a pair of atoms and has one of 8 types.

A ten fold cross validation experiment was carried out and gave an average accuracy of 87.3% with a standard deviation of 4.99%. This is comparable to the results obtained on the same data by others [6].

Unfortunately, the induced theories seem to rely rather heavily on the additional properties of the molecules (i.e., `Ind1, IndA` and `Lumo`) rather than their molecular structure. To further test the value of the decomposition technique, we intend to repeat this problem with only the molecular structure to describe the examples (i.e., leaving the three "propositional properties" out).

5 Conclusion and Future Work

A new approach to dealing with highly structured examples has been presented. This approach relies on a technique called decomposition that is able to extract the structural predicates from an example, thus allowing the learning system to concentrate on the properties that the components of such structure may present. A learning system, ALFIE, based on this idea has been implemented and preliminary experiments demonstrate promise.

A number of issues remain open as the subject of future work. In particular,

- ALFIE, like other similar learning systems, is susceptible to the order of the examples. In ALFIE's case, it is interesting to point out that the effect of ordering seems more evident in problems whose examples have little or no structure.
- It is possible to have more operators for the construction of the E_i's, such as negation and disjunction.
- Decomposition can be improved further by analysing the sets that are produced. It is likely that information is being repeated. This should be easy to check and correct. It would also be interesting to see how the depth of the decomposition affects the accuracy of the learning system.

Finally, more experiments are needed with problems presenting high structures in their examples and the associated concepts.

References

1. M. Bongard. *Pattern Recognition*. Spartan Books, 1970.
2. A.F. Bowers. Early experiments with a higher-order decision-tree learner. In *Proceedings of the COMPULOGNet Area Meeting on Computational Logic and Machine Learning*, pages 42–48, 1998.
3. A.F. Bowers, C. Giraud-Carrier, C. Kennedy, J.W. Lloyd, and R. MacKinney-Romero. A framework for higher-order inductive machine learning. Compulog Net meeting, September 1997.
4. A.F. Bowers, C. Giraud-Carrier, and J.W. Lloyd. Higher-order logic for knowledge representation in inductive learning. 1999, (in preparation).
5. P.A. Flach, C. Giraud-Carrier, and J.W. Lloyd. Strongly typed inductive concept learning. In *Inductive Logic Programming: ILP-98*, 1998.
6. R.D. King, S. Muggleton, A. Srinivasan, and M. Sternberg. Structure-activity relationships derived by machine learning: The use of atoms and bonds and their connectivities to predict mutagenicity in inductive learning programming. *Proceedings of the National Academy of Sciences*, 93:438–442, 1996.
7. N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.
8. J.W. Lloyd. Declarative programming in Escher. Technical report, University of Bristol, 1995.
9. S. Muggleton, editor. *Inductive Logic Programming*. Academic Press Ltd., 24-28 Oval Road, London NW1 7DX, 1992.
10. C. Rouveirol. *Flattening and Saturation: Two Representations Changes for Generalisation*, volume 14. Kluwer Academic Publishers, Boston, 1994.