Learning to Classify X-Ray Images Using Relational Learning

Claude Sammut[1] and Tatjana Zrimec[1,2]

[1] School of Computer Science and Engineering, University of New South Wales, Sydney 2052, Australia
[2] Faculty of Computer and Information Science, University of Ljubljana, Tržaška 25, 1001 Ljubljana, Slovenia

Abstract: Image understanding often requires extensive background knowledge. The problem addressed in this paper is such knowledge can be acquired. We discuss how relational machine learning methods can be used to automatically build rules for classifying types of blood vessels. We introduce a new learning system that can make use of background knowledge coded as arbitrarily complex Prolog programs to construct concept descriptions, particularly those needed to classify features in an image.

1 Introduction

Model-based image processing is the application of knowledge about objects expected in a scene to the recognition of those objects if they appear in an image. In medical image understanding (Zrimec & Sammut 1997; Robinson *et al* 1993), expert radiologists bring a wealth of experience to bear on the problem of interpreting x-ray images. Much of the knowledge needed for such a task can be obtained from text books. However, each radiologist accumulates a large amount of personal experience in understanding the contents of an x-ray image. The challenge for an automated image understanding system is how such experience can be gathered by a program. In this paper, we argue that a relational learning system can acquire such knowledge. However, the program itself must also be capable of using extensive background knowledge.

We introduce a new learning system that extends Cohen's (1996) work on refinement rules. This system, which is part of the *iProlog* machine learning environment (Sammut 1997), can take advantage of background knowledge encoded as arbitrarily complex Prolog programs. As we shall see, this is particularly useful in constructing some of the high-level relations needed to synthesise programs capable of recognising different types of blood vessels.

A further problem for learning in this domain is that while the examples are quite complex, there are only a few of them. This is due to the difficulty of obtaining x-ray images that have been fully interpreted and labelled by an expert. Because of the small sample size, we employ a specific-to-general search based on Plotkin's (1971) relative least general generalisation (RLGG). However, we severely limit the number

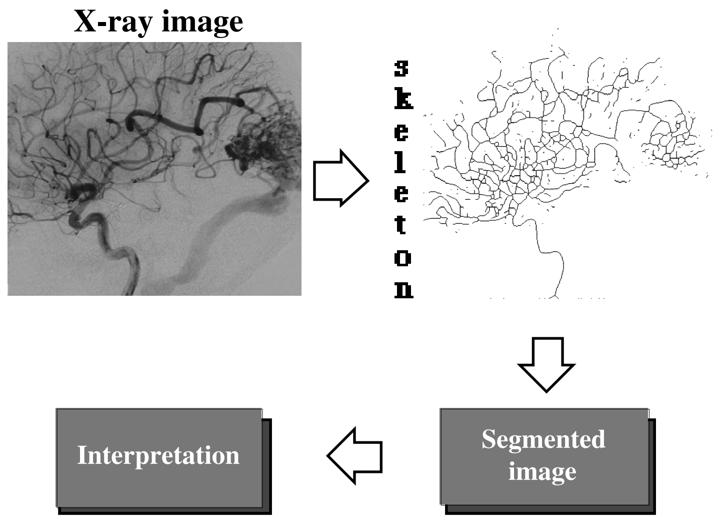

Fig.1. The image interpretation process

of literals in the RLGG in a manner related to those described by Page and Frisch (1992).

In the following section, we briefly describe the image processing required to obtain the input to the learning program. We then introduce the mechanisms for generating clausal descriptions of the examples and the generalisation mechanism. We conclude with a discussion of the current status of the work and future directions.

2 Interpreting X-ray Images

Figure 1 outlines the process of interpretating x-ray angiograms of a patient's cerebral vasculature. X-rays are normally taken from several standard views of the patient's brain. An image passes through the following preprocessing stages.

1. The grey-scale x-ray image is thresholded to obtain a black-and-white image.
2. The black-and-white image is *skeletonised* to reduce thick vessels to lines only a single pixel wide.
3. The skeleton is traced to join pixels into segments of blood vessels.
4. The segmented skeleton is used to guide further processing of the grey-scale image to obtain diameters and intensity values of each blood vessel segment.

In practice, this process is iterative. Different levels of thresholding reveal more detail and also more noise. The method used here is to first apply aggressive thresholding, obtaining only the most prominent blood vessels. Gradually, the threshold is decreased to admit more detail. The blood vessels recognised in the previous pass can be used to guide further recognition. For the purposes of this paper, we will only consider a single pass with a high threshold to obtain a relatively simple set of segments.

The output of the tracing program is a set of Prolog facts that will be input to the learning program. For example, the following clauses:

```
blood_vessel(mb1, 1, 'ICA').
segment(1, mb1, n, 40, 130, [2]).
segment(2, mb1, w, 40, 144, [3]).
segment(3, mb1, nw, 35, 135, [4, 5]).
segment(4, mb1, n, 40, 50, [6, 7]).
segment(6, mb1, ne, 20, 170, [8, 9]).
segment(5, mb1b1, e, 10, 100, []).
segment(7, mb1b2, w, 5, 125, []).
segment(8, mb1b3, e, 18, 90, []).
segment(9, mb1b4, n, 15, 100, []).
```

describe a blood vessel *mb1*, of type *ICA* (Internal Carotid Artery) which starts with segment number 1. Each segment is described by an atom with the following arguments: the segment's segment number, the identifier of the blood vessel to which the segment belongs, the direction of the segment (north, north-east, east, south-east, *etc*), the diameter of the segment the grey-level intensity of the segment and finally, a list of segments that branch from the end of this segment. A segment's boundaries are where there is a branch or a bend in the blood vessel.

Suppose we wish to train the learning program to recognise different types of blood vessels. What kinds regularities would we expect the program to discover from examples of, say, the Internal Carotid Artery? One may be that the diameter reaches its maximum somewhere near the middle segment and this always corresponds to the intensity at its lowest value. Another may be that the final segment always ends pointing north-east. And another is that the internal carotid artery always has branches leading to two other arteries before it bifurcates into two more arteries, one of which goes north and the other east. The learning system must invoke background knowledge to obtain a useful set of predicates for such descriptions. In the following section, we describe how we make use of background knowledge.

3 Refinement Rules and Generalisation

Cohen (1996) introduced refinement rules as a method for constructing new literals to add to clauses during a general-to-specific search. A restricted second order theorem prover was is to interpret these rules. However, this theorem prover is limited to a simple function-free language. The system described in this paper is a component of *iProlog* (Sammut 1997). This is an ISO compatible Prolog interpreter with a variety of machine learning tools embedded as built-in predicates. Since the full power of Prolog is available, the refinement rules we implement can invoke arbitrary Prolog programs.

Two types of refinement rule must be defined. A *head rule* has the form:

$$\langle A, Pre, Post \rangle$$

where *A* is a positive literal, *Pre* is a conjunction of literals and *Post* is a set of positive literals. A *body rule* has the form:

$$\langle \leftarrow B, Pre, Post \rangle$$

where *B* is a positive literal and *Pre* and *Post* are as above.

There must only be one head rule to that *A* should be used to create the head of the clause being learned, provided that the condition *Pre* is satisfied. After *A* has been constructed, the literals in *Post*, are asserted into Prolog's database. There may be any number of body rules to generate literals for the body of the clause under construction. Literals in the precondition of these rules may invoke any Prolog program.

Suppose we wish to create a saturated clause (Rouveirol & Puget 1990; Sammut 1986) based on the example of blood vessel *mb1,* shown in the previous section. The left-hand side of the following rule is the template for the head literal.

```
blood_vessel(VesselName, StartingSegment, VesselType)
    where
            true
    asserting
            seg_list(VesselName, [StartingSegment]).
```

The *where* part of the rule is the precondition and the *asserting* part is the post-condition. Refinement rules are invoked in a forward chaining manner. The head rule matches the predicate blood_vessel(mb1, 1, 'ICA'). Since there are no preconditions, the head of the new clause is created and the predicate seg_list(mb1, [1]) is asserted into the database. This enables the following body rule to construct literals for the segments of the blood vessel:

```
(:- segment(SegId, VesselName, Dirn, Diam, Inten, SegList))
    where
            seg_list(VesselName, S),
            member(SegId, S)
    asserting
            seg_list(VesselName, SegList),
            diameter(VesselName, SegId, Diameter),
            intensity(VesselName, SegId, Intensity).
```

For each match of the template and for each solution to the preconditions, a new segment literal is created and the corresponding post-conditions are asserted. After execution of this rule, the clause is:

```
blood_vessel(mb1, 1, 'ICA') :-
        segment(1, mb1, n, 40, 130, [2]),
        segment(2, mb1, w, 40, 144, [3]),
        segment(3, mb1, nw, 35, 135, [4, 5]),
        segment(4, mb1, n, 40, 50, [6, 7]),
        segment(6, mb1, ne, 20, 170, [8, 9]).
```

We next include a rule that constructs a '>' relation on the diameters and intensities of the blood vessel segments.

```
 (:- X > Y)
    where
           measurement(M), M(V, S1, X), M(V, S2, Y).

measurement(distance).
measurement(intensity).
```

A second order extension to Prolog permits the principal functor of a predicate to be a variable provided that at run time, the variable is bound to a valid predicate symbol. Without the precondition above, it would be possible to have comparisons where one argument is a diameter and another is an intensity or between numbers belonging to different blood vessels. We can now see that the assertions in the *segment* rule provide type information used by the X > Y rule. Execution of this rule would add literals of the form X > Y for all pairs of diameters and intensities of segments in *mb1*.

A more sophisticated piece of background knowledge is required if we wish to include in the concept description which segment has, say, the maximum diameter. This is a little tricky because the refinement rule must scan all of the segments to find the maximum value. We now see the power of Prolog being used to construct a new literal. First, we require a predicate that can find the maximum value of a measurement. We use the second order extension to make this predicate generic for different types.

```
max(M, V, S, X) :-
    findall(M(V, S, N), M(V, S, N), L),
    max(L, M(V, S, X)).

max([X], X) :- !.
max([M(V, SA, A)|B], M(V, S, X)) :-
    max(B, M(V, SY, Y)),
    (A > Y -> S = SA, X = A; S = SY, X = Y).
```

This can be read as: the maximum value of measurement *M* in blood vessel, *V*, occurs in segment, *S*, and has value, *N*. The predicate can be invoked from the body rule:

```
 (:- max(M, V, S, N))
    where
           measurement(M),
           blood_vessel(V, _, _),
           max(M, V, S, N).
```

A corresponding rule can defined for the minimum value. After execution of these rules, the following literals are added to the clause:

```
max(diameter, mb1, 1, 40)
max(diameter, mb1, 2, 40)
max(diameter, mb1, 4, 40)
max(intensity, mb1, 6, 170)
min(diameter, mb1, 6, 20)
min(intensity, mb1, 4, 50)
```

The segment of interest is number 4 since it has the maximum diameter and minimum intensity.

Once a clause has been saturated, we need a generalisation mechanism. We mentioned earlier that while the examples in this domain can give rise to quite complex descriptions, only small data sets are available. Typically, general-to-specific search methods require a reasonably large sample for their statistics to be accurate. For this reason, we have chosen to experiment with a specific-to-general search based on Plotkin's (1971) relative least general generalisation (RLGG). The first step is to use saturation, as described in the previous section, to build clauses which are then passed to an LGG algorithm for generalisation. However, a pure LGG generates far too many irrelevant literals. Therefore, we follow an approach similar in spirit to the constrained atoms of Page and Frisch (1992). Since the refinement rules impose restrictions on the form of literals that can be generated through saturation, it is reasonable to apply the same restrictions to literals constructed by generalisation. Thus, we modify Plotkin's LGG algorithm to filter literals so that whenever an LGG of two literals is found, it is tested against the refinement rules. If no refinement rule is satisfied, the LGG is rejected. As a result, the RLGG's do not grow to impractical sizes.

4 Discussion

The current status of this project is that the programming of the image processing and machine learning software has been completed. Initial testing on sample x-ray images is has begun, but full-scale trials are yet to be conducted. There are several novel aspects to this work:

- We have modified Cohen's refinement rule approach to permit the introduction of complex background knowledge through Prolog programs.
- We have used the refinement rules, together with a method for tagging literals to constrain the size of RLGG's.
- We have applied this approach to the problem of model-based image interpretation, particularly for x-ray images.

We believe that studying methods for making effective use of intentional background knowledge is important for the development of ILP. It's greatest advantage over propositional learning methods is the possibility of employing background knowledge to construct concept descriptions that are beyond the capabilities of other kinds of learning systems.

References

Cohen, W. (1996). Learning to classify English text with ILP methods. In L. D. Raedt (Eds.), *Advances in Logic programming.* (pp. 124-143). IOS Press.

Page, C. D., & Frisch, A. M. (1992). Generalization and Learnability: A study of constrained atoms. In S. Muggleton (Eds.), *Inductive Logic Programming.* (pp. 29-61). Academic Press.

Plotkin, G. D. (1971). A further note on inductive generalization. In B. Meltzer & D. Michie (Eds.), *Machine Intelligence 6.* New York: Elsevier.

Robinson, G.P., Colchester , A.C.F., Griffin, L.D. (1993) Model Based Recognition of Anatomical Objects from Medical Images. In *Information Processing in Medical Imaging*, 13th International Conference, IPMI'93, Arizona, USA, (pp. 197–211).

Rouveirol, C., & Puget, J.-F. (1990). Beyond Inversion of Resolution. In *Proceedings of the Seventh International Conference on Machine Learning*, Morgan Kaufmann.

Sammut, C. A., & Banerji, R. B. (1986). Learning Concepts by Asking Questions. In R. S. Michalski Carbonell, J.G. and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach, Vol 2.* (pp. 167-192). Los Altos, California: Morgan Kaufmann.

Sammut, C. (1997). Using background knowledge to build multistrategy learners. *Machine Learning*, **27** , 241-257.

Zrimec, T. and Sammut, C.A., (1997). A Medical Image Understanding System, *Engineering applications of Artificial Intelligence,* February, **10** (1), 31-39.