Selective Inductive Logic Programming

Heather Maclaren

Dept. of Computer Science, University of York,
Heslington, York, YO10 5DD, UK
heather@cs.york.ac.uk

Abstract. Given a target theory to learn that will continuously change over time, an Inductive Logic Programming (ILP) algorithm would have to constantly revise its current hypothesis. Instance-based learning methods step round the need for theory revision, but don't allow the same modelling of the underlying theory that we can accomplish using standard ILP. In addition, the representations of the concepts modelled by instance based-learning are not easily understood. We present a method for learning a target theory that changes over time, combining Inductive Logic Programming with instance based learning to produce a framework which should draw on the positive aspects of both methods.

1 Introduction

There are several areas of possible learning where the theory to be learnt is not 'clear cut', i.e. where there may not be a final target theory, or the theory may even change over time. One such area is modelling human actions in an attempt to predict what they will do if given a particular situation. For example, in a diary management system, we may wish to predict whether or not the user will accept a meeting request. This could depend on the date, time, nature of meeting, who the meeting is with, what other work the user has to do at this point, or other personal things such as a family birthday. All of these factors will change over time and affect how the user will react to decisions that could arise.

ILP is useful in this area because we can use items of background information about the user and because it attempts to model the underlying relationships between items of data to work out why they are carrying out these actions given this particular state. Thus, given a new state we can use our knowledge of the user to predict what they will do. We can also incorporate some intensional background knowledge, in this case 'rules of thumb' about what humans tend to do. For example, the user cannot be in two places at once or transport simultaneously from one place to another (unless of course they are adjoining rooms).

But humans can change their habits or patterns of actions without warning and for reasons which we are unable to detect which gives their actions a seemingly random rate and direction of change (for example the user may acquire a puppy and have to go home every lunchtime for a few weeks, the only things we would see would be the effects - longer lunchbreaks and refusal of all working lunches). If we were going to approach this problem using purely ILP then we would need to include constant theory revision and the uncertainty of the theory we are attempting to learn changing over time as all rules learnt would have the potential to become unsound. In addition there may be only a few examples, which would reduce our ability to learn rules with any acceptable degree of accuracy.

Lazy instance-based learning methods effectively step round the problem of theory revision as they have no rules to revise, they simply make use of all the examples in the surrounding area whenever a query is received. In addition a lazy instance-based method would be able to accommodate the theory being modelled changing by 'forgetting' some of the oldest examples. Plus, while it doesn't increase the accuracy of predictions if we have small numbers, we can start to attempt classifications earlier than ILP as to attempt to create rules by generalising over insufficient data will not create anything of use.

Some of the problems with instance-based learning are that the representations of the concepts captured are not easily understood, and that we cannot use intensional background knowledge. Both of these problems are overcome by ILP. If we were to combine the ideas from ILP and an instance based method such as case-based reasoning (CBR) then we would be able to model the relationships between items of data in a comprehensible fashion, use intensional background knowledge, and in addition be able to cope more effectively with a moving target theory.

Selective ILP only learns rules to cover positive example clusters that satisfy a given set of criteria. All other examples are left uncovered but are still used when answering queries using case-based reasoning. The aim is to produce a few 'good quality' rules which should require less frequent revision rather than lots of average or poor quality rules that are in need of being constantly updated.

The structure of this paper is as follows. Section 2 outlines the theory on which this new method is based, section 3 describes the formulation of the framework of the method, and section 4 outlines areas of possible further work.

2 Background Theory

The problem of inductive inference is as follows. Given a background (prior) knowledge B and evidence E. The evidence $E = E^+ \wedge E^-$ consists of positive evidence E^+ and negative evidence E^-. The aim is then to find a hypothesis H such that the following conditions hold.

Prior Satisfiability. $B \wedge E^- \not\models \square$
Posterior Satisfiability. $B \wedge H \wedge E^- \not\models \square$
Prior Necessity. $B \not\models E^+$
Posterior Sufficiency. $B \wedge H \models E^+$

(Muggleton,De Raedt 94 [5])

Inductive Logic Programming [5] is a method for generating a theory in the form of a logic program, from a set of examples. The set of relations produced are an attempt to capture the underlying relations between the items of data. Background knowledge in the form of relations or ground facts may be used in the construction of these relations.

(Simple) instance-based learning [9] stores each example and when a new item is to be classified its relationship to the existing examples is examined to see which of them it is the most similar to. The new item is then given the same classification as the existing example it was closest to. Usually it is not just the nearest example that is taken into account, but some number of the nearest neighbours. The most common classification amongst those instances is returned as the classification of the query. There are several approaches to measuring distances between first-order objects, recent ones include [3],[6] and [8].

We will want to learn some rules from the data to make query answering faster and to aid in explaining our actions to the user if they ask. But although rules are quick to return answers, they take time to learn so we only want to learn those rules which are of the 'highest quality'. i.e. if we get a group of similar positive examples in the instance space then we want to learn a rule that covers that group, but we don't want to worry about covering solitary examples. Hence we need to investigate some form of clustering.

A possible candidate for a clustering algorithm would be the Leader algorithm [2]. A basic description of the algorithm is: Given a distance measure and a threshold, the Leader algorithm will go though each example and work out which leader of the existing clusters the example is closest to. If the distance between the example and its nearest leader is below the given threshold then the example is assigned to that group. If not, then the example is put into a new cluster and made the leader of that cluster.

One of the major advantages of this algorithm is speed, it only deals with each example once and only compares it to the leader of each existing group rather than to every other example. This reduces the number of comparisons made to $\sum_{i=1}^{M} n_i$, where M is the number of examples being assigned and n_i is the number of clusters in existence when we are attempting to assign the i'th example. The *maximum* number of comparisons will be $\sum_{m=1}^{M}(m+l)$, where l is the number of clusters already in existence before we started to assign examples, this would occur when each example is forced into a new cluster. However, given the nature of leader choice, the leader of each cluster may not be in the best position to create the best clusters. Therefore it may be an idea to create a cluster centre after more than one example is assigned to a cluster and use that as the leader, re-calculating it when another example is assigned. It will add to the computation time but will create better clusters. The computation time will be $M-1+\sum_{i=1}^{M} n_i$, at *most* $M-1+\sum_{m=1}^{M}(m+l)$ (which is equivalent to $(M^2+M(l+3)-2)/2$) compared to a *constant* computation time of M^2 for a simple agglomerative clustering algorithm.

After we have learnt a few select rules to help in prediction of actions, and explanation of our actions to the user, it may be the case that as new examples occur these rules become inaccurate. It would be inefficient to simply remove each rule that became inaccurate, so some attempt at theory revision should be made to see if the rule can be salvaged. By generalising the rule we can make it cover any more positive examples that may reside in the space next to the cluster that the rule captures. By specialising the rule we can attempt to exclude some of the negative examples that have occurred since the rule was learnt. Hence combining these two we can hopefully shift the area covered by the rule to match the shift in theory that has occurred since the rule was learnt. Ideas from theory revision systems such as FORTE [7] and CLINT [1] may be useful in this area.

3 Selective ILP

In this section we outline the proposed approach to combining ILP and CBR. All undefined variables which occur in the material presented are intended to be constants, with values which should be provided based on the context in which the approach is applied to a learning problem (i.e. provided by the user). Appropriate default values may be determined later on during experimentation and empirical evaluation.

To construct the algorithm let us start with the sample space. As we are dealing with ILP, let us assume that each item will be a positive or negative example of the predicate we are trying to learn.

Each example is represented as a unique fact (i.e. a ground unit clause). We will use some element of clustering to determine when a set of examples is suitable to generalise over, and then use some form of ILP algorithm to do the learning.

To determine suitable clusters, the clustering algorithm described earlier will be used on the examples in the sample space to produce some rough clusters, then clusters will be picked that satisfy the following criteria:

1. The cluster must have at least j positive elements.
2. At least j/i elements must be uncovered examples.
3. The cluster must satisfy the evaluation threshold e.

Evaluation of rules and potential clusters will use the simple inequation $\frac{p}{p+n} > e$ where p is the number of positive examples covered n is the number of negative examples covered, and e is the minimum evaluation score allowed.

The Leader algorithm has two requirements. Firstly it needs a distance measure with which to compare the instances, and secondly it needs a threshold to compare to the distance between each example and its nearest leader. Both of these requirements will require further thought and experimentation. Distance measures between examples must include some form of semantic comparison as ILP is concerned with learning the underlying relationships between items of data rather than just recursively comparing terms. This is an important consideration, but one that has not yet been fully investigated with regards to this application. Once the nature of the distance measure has been established we can then work out some sort of threshold as this will rely on the nature of the distance measure used.

Once we have our clusters to learn from, we will then proceed to use some ILP method on each cluster in turn. We can use the closed world assumption of ILP to help in this learning by letting the ILP algorithm think that all known instances that are not in the cluster it has been given are negative instances, thus forming a boundary for the space which the hypothesis can cover. Which ILP method to use is as yet undecided, the criteria is a method which can produce a hypothesis with as few rules as possible which takes up as little extra instance space as possible outside the perimeter of the area occupied by the cluster. The reason for this last criterion is that if we are to produce hypotheses which do not require frequent updating, then we should try to avoid covering unknown areas of sample space outside the perimeter of the cluster incase these are later found to contain significant numbers of negative examples. All examples not covered by a rule will be covered (in a sense) by CBR, so we do not lose any information.

Once we have fed our clusters to the ILP algorithm we will end up with a sample space that contains a mixture of areas covered by rules and individual examples. See figure 1.

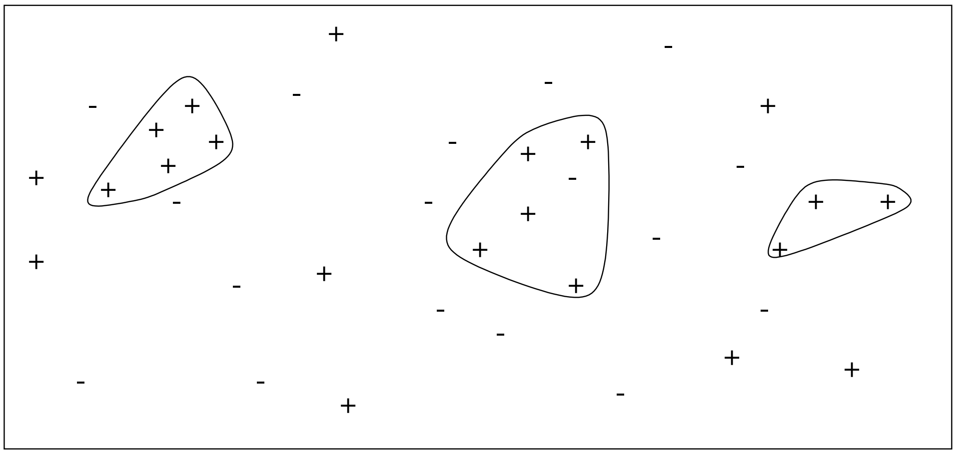

Fig. 1. The sample space. + denotes a positive example, - denotes a negative example.

The rules created will fire with much greater speed than a CBR query. In this context as the rules created by the ILP algorithm are part of a logic program a rule will 'fire' if it evaluates to true with all its arguments instantiated by the time it has been fully evaluated (whether or not they were all instantiated when it was called). So when a query arrives, we will check to see if a rule covers it before resorting to using CBR. The coversets of rules may have an overlap, and in this case we will use the majority vote, with each rule's answer begin weighted by the evaluation score of the rule.

Our algorithm is as follows:

Learning Phase

For each example in the queue:

Add it to the sample space. Does a rule fire?
    Yes: Is the example positive?
        Yes: Increase the rule's positive cover set count.
        No: Increase the rule's negative cover set count.
(An up to date record of each rule's evaluation score is needed in case of a vote in the query phase)

Every x examples:

– Housecleaning. Remove the oldest l examples. $l = \frac{x^2}{x+c}$
– Attempt to revise each rule that now fails the evaluation test $e <= \frac{p}{p+n}$
  If the rule is still unable to pass the test after revision, or its coverset is less than q in size, remove it.
– Have a look to see if a new rule can be learnt. Use clustering to produce a set of candidate clusters, select all those with at least j positive examples and at least $\frac{j}{i}$ of these examples are uncovered. Feed each successful cluster to the ILP algorithm to create a new hypothesis.

Query Phase

For each query:

Add it to the sample space, does a rule return an answer?
    Yes: Does more than one rule return an answer?
        No: Return the answer given.
        Yes: Return a majority vote with answers weighted by each rule's evaluation score.
        If no suitable majority then use CBR.
    No: use CBR.

If CBR cannot produce a suitable answer then we cannot answer this query.

What if a rule produces more than one answer? The rules created by the ILP method to cover a cluster may not be deterministic. In this case we will have to resort to CBR as we have no further means of distinguishing between the possible answers.

4 Further Work

The algorithm outlined earlier is definitely still in its early stages and we have a number of items still to consider. As more examples are added it may be that the coversets of some rules will overlap. If this happens above a certain extent and the rules concerned provide the same answer then we should consider merging them.

Learning of recursive predicates may present problems if we are continuously updating and revising our rule-base, the same would apply to learning multiple predicates. One final extension that should be possible would be to learn from positive examples only. As we are using a form of clustering to generate candidates for rule learning this should not be a problem, although the method of rule evaluation would need alteration.

A method of combining ILP and instance-based learning has already been investigated in Analogical Prediction [4], however this still uses the closed world assumption of ILP, hence all instances that need to be classified as positive must be covered by some rule. The method outlined here draws on the proximity measures of instance-based learning to remove that assumption, thus reducing the number of rules required and catering for a constantly updating instance space without having to relearn or revise the theory after each example is added.

5 Acknowledgements

Thanks are due to James Cussens, Sara Farmer, and Chris Kilgour for pointing me in the right direction every so often. This work is support by BT EPSRC CASE award No. ML825187/CT381735.

References

1. L. De Raedt. *Interactive Theory Revision: An Inductive Logic Programming Approach.* Academic Press, 1992.
2. John A. Hartigan. *Clustering Algorithms*, chapter 3. John Wiley & Sons, 1975.
3. M. Kirsten and S. Wrobel. Relational distance-based clustering. In D. Page, editor, *Proceedings of the 8th International Conference on Inductive Logic Programming*, volume 1446 of *Lecture Notes in Artificial Intelligence*, pages 261–270. Springer-Verlag, 1998.
4. S. Muggleton and M. Bain. Analogical prediction. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 234–244. Springer-Verlag, 1999.
5. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.
6. J. Ramon and M. Bruynooghe. A framework for defining distances between first-order logic objects. In D. Page, editor, *Proceedings of the 8th International Conference on Inductive Logic Programming*, volume 1446 of *Lecture Notes in Artificial Intelligence*, pages 271–280. Springer-Verlag, 1998.

7. B.L. Richards and R.J. Mooney. First-order theory revision. In L. Birnbaum and G. Collins, editors, *Proceedings of the 8th International Workshop on Machine Learning*, pages 447–451. Morgan Kaufmann, 1991.
8. M. Sebag. Distance induction in first order logic. In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297 of *Lecture Notes in Artificial Intelligence*, pages 264–272. Springer-Verlag, 1997.
9. D. Michie D.J. Spiegelhalter and C.C. Taylor, editors. *Machine Learning, Neural and Statistical Classification*, chapter 4. Ellis Horwood, 1994.