Learning Minor Closed Graph Classes with Membership and Equivalence Queries

John Shawe-Taylor
Dept of Computer Science,
Royal Holloway, U. of London[1]
Email: john@dcs.rhbnc.ac.uk

Carlos Domingo, [2]
Department of Software,
U. Politècnica de Catalunya[3]
Email: carlos@lsi.upc.es

Hans Bodlaender,
Dept of Computer Science,
Utrecht University[4]
Email: hansb@cs.ruu.nl

James Abello
Computer Science Dept,
Texas A&M University[5]
Email: abello@cs.tamu.edu

NeuroCOLT Technical Report Series

NC-TR-94-014

January, 1995[6]

Produced as part of the ESPRIT Working Group
in Neural and Computational Learning,
NeuroCOLT 8556

[1]Egham, Surrey TW20 0EX England

[2]Author supported by the Esprit EC program under project 7141 (ALCOM-II) and Working Group 8556 (NeuroColt), and by the Spanish DGICYT (project PB92-0709).

[3]Edifici FIB, Pau Gargallo 5 08028 Barcelona, Spain

[4]3508 TB Utrecht, The Netherlands

[5]College Station, TX 77843, USA

[6]Received November 5, 1994

Abstract

The paper considers the problem of learning classes of graphs closed under taking minors. It is shown that any such class can be properly learned in polynomial time using membership and equivalence queries. The representation of the class is in terms of a set of minimal excluded minors (obstruction set).

1 Introduction

This paper considers the problem of identifying a very broad series of classes of graphs, namely those closed under taking graph minors. Such sets of graphs have been studied for a number of years by graph theorists, most notably in a series of papers by Robertson and Seymour (see for example among other papers [10, 12, 11, 14, 13]). The classes of graphs that are closed under taking minors are very common. Examples are the planar graphs, graphs which can be embedded in 3-dimensional space without knots, graphs which can be embedded in 3-dimensional space without interlocking cycles, graphs with genus, treewidth, pathwidth at most some fixed constant k, etc. In some cases there are no known algorithms for testing for membership in these classes, though the general theory of graph minors ensures that such algorithms must exist. One way of viewing our results is a general method of learning such algorithms.

The key result obtained by Robertson and Seymour showed that for any class of graphs closed under taking minors, there is a *finite* set of minimal minors not in the class. This set is called the obstruction set of the class. Any such class can therefore be characterised as the set of graphs which do not contain any member of the obstruction set as a minor. For the example of planar graphs the obstruction set consists of the two non-planar graphs, K_5 and $K_{3,3}$. Hence, in this case the result is equivalent to the famous Kuratowski theorem that a graph is planar if and only if it does not contain one of these two graphs as a minor. In general there are no efficient algorithms for computing the obstruction set of a class of graphs, and in some cases of interest (e.g. the set of graphs embeddable in 3-space such that no cycle forms a knot) the obstruction set is not yet known.

This paper shows that any graph class that is closed under taking minors can be identified using the learning protocol of equivalence and membership queries. The learning algorithm delivers the obstruction set of the class using a number of queries that is polynomial in the size of the minimal representation and the size of the largest counterexample.

Once the obstruction set has been constructed, there are well developed algorithms for deciding whether a given graph has a member of the obstruction set as a minor. These algorithms are at worst cubic in the number of vertices in the graph, but exponential in the size of the obstruction set. Hence, the representation fails to be polynomially evaluable in its own size but, once it is fixed, is polynomially evaluable in the size of the graph which is being evaluated. This technical point will not be relevant to our learning algorithm, since we will not need to evaluate the representations on any examples. In summary, our learning algorithm uses membership and equivalence queries to "learn" an algorithm that is known to exist, but which we do know how to build.

The notion of query learning was introduced by Angluin in 1987 [1]. In that paper she proved the first and strong positive result, namely that the class of regular languages represented by deterministic finite automata are learnable in polynomial time using membership and equivalence queries. Many other positive results have been proved in this model. For instance, restricted classes of grammars, restricted classes of boolean formulas, some geometrical concepts, etc are query learnable.

On the other hand, there exists many negative results either in terms of the number of queries needed or the set of queries allowed. The paper [3] introduced the concept of an approximate fingerprint, which provides a general technique for proving negative results in query learning. More recently, there has been an increasing number of negative results proved using tools from the field of structural complexity. All these results can be found in the recent surveys [4, 5].

Our algorithm is interesting in the sense that it uses the queries to produce a representation for a concept that we do not know how to build otherwise, even if we are given an algorihm that effectively decides the concept.

In the next section we will review the definitions and results of the theory of graph minors and of learning theory that we will need. This will lead into Section 3 containing our main results. Finally, the conclusion indicates how the results can be generalised and highlights questions that remain unresolved.

2 Definitions and Known Results

A graph G comprises a set of nodes VG and a set of edges $EG \subseteq VG \times VG$, which is symmetric and antireflexive. We say that a graph H is a one-step minor of a graph G, denoted $H \prec_1 G$, if H is obtained from G by one of the following operations:

1. deletion of one edge;
2. by deletion of one vertex together with all edges incident with the vertex; or
3. by identifying two adjacent vertices into a single vertex, that is adjacent to all the vertices adjacent to either of the two identified vertices.

The minor relation (denoted $\preceq$) is the transitive reflexive closure of the relation $\prec_1$.

A class of graphs $\mathcal{G}$ is minor closed if $G \in \mathcal{G}$ and $H \preceq G$ imply that $H \in \mathcal{G}$. A graph property P is minor closed if the class of graphs having property P is minor closed.

For a minor closed class of graphs $\mathcal{G}$, the obstruction set of $\mathcal{G}$, denoted $\mathrm{ob}(\mathcal{G})$, is the set of minimal elements in the relation $\preceq$ in the complement of $\mathcal{G}$. Hence, for each graph G, $G \in \mathcal{G}$ if and only if there is no $H \in \mathrm{ob}(\mathcal{G})$ that is a minor of G.

Robertson and Seymour proved Wagner's conjecture that for every minor closed class of graphs, the obstruction set is finite. Their theorem states.

Theorem 1 *[7] For every minor closed class of graphs $\mathcal{G}$ the obstruction set $ob(\mathcal{G})$ of $\mathcal{G}$ is finite.*

Corollary 2 *Every minor closed class of graphs is decidable.*

In fact the situation is much better than being simply decidable. Efficient algorithms have been devised for testing whether a fixed given graph is a minor of a graph G, which are polynomial in $|G|$.

Theorem 3 *[7] For every graph H, there exists an $O(n^3)$ algorithm, that given a graph G, tests whether H is a minor of G.*

Corollary 4 *Given the obstruction set ob($\mathcal{G}$) of a minor closed class of graphs $\mathcal{G}$, we can test whether a graph G is a member of the class in $O(|G|^3)$ time.*

In the case of classes that do not contain all planar graphs, this complexity can be improved using results about graphs of bounded treewidth [7].

Proposition 5 *Given the obstruction set ob($\mathcal{G}$) of a minor closed class of graphs $\mathcal{G}$ that does not contain all planar graphs, we can test whether a graph G is a member of the class in $O(|G|)$ time.*

In this paper we follow the learning framework defined by Watanabe [15] and extended in [16]. In order to specify our learning problem we have to determine both a concept space and a way of representing the concepts. The formal object known as a *representation class* is, informally, a set of valid representations R, a semantic function from R to the concept space and a size function to measure a given representation. We will not give more details here since our work is centered in one particular representation class.

Throughout the paper we will work with the class of concepts $\mathcal{G}$ of graphs closed under taking minors. We can represent a class $\mathcal{G}$ in two ways:

- ◇ by a finite set of minimal excluded minors. We call this representation class R_{ob};
- ◇ by a formula in Monadic Second Order Logic. We denote this representation class R_{MSOL} [7].

Clearly, both representation are mapped into the concept class $\mathcal{G}$ but the second one appears to be much more powerful. In our algorithm we will work with the first one. The representation of the classes of graphs closed under taking minors will thus be in terms of their obstruction set. Hence, we consider the finite sets of graphs which form an antichain in the 'is a minor' relation. The corresponding class of graphs is obtained by taking the complement of the set of graphs which have at least one graph from the set as a minor.

Further, we will denote by ob($\mathcal{G}$) $\in R_{\text{ob}}$ the target concept that our Learner is trying to discover.

When we are talking about query learning, we need to define the communication protocol between the *Teacher* and the *Learner*. A protocol is a set of queries. We will use just one protocol, which is the most common one. Our protocol has two queries, equivalence and membership, defined as follows.

- ◇ An **equivalence** query presents an `Obstruct` $\in R_{\text{ob}}$ (which we call a "conjecture") to the Teacher who replies "True" if it is equivalent to the target concept ob($\mathcal{G}$), or "False", together with a counterexample, i.e. a graph which is classified differently by `Obstruct` and ob($\mathcal{G}$).
- ◇ A **membership** query presents a graph G to the Teacher, who indicates whether or not G is in the concept represented by ob($\mathcal{G}$).

Thus, we will say that the class R_{ob} is learnable in polynomial time using membership and equivalence queries if there exists an algorithm S that learns any ob(G) $\in R_{\text{ob}}$ from any Teacher T in time bounded polynomially in the size of the representation and the length of the longest counterexample given by T.

3 Learning minor closed classes

The main result of this paper can now be stated.

Theorem 6 *Let $\mathcal{G}$ be a minor closed class of graphs. The obstruction set of $\mathcal{G}$ can be constructed in polynomial time using membership queries bounded by the size of the largest counterexample and one equivalence query for each graph in the obstruction set.*

Proof: The algorithm which identifies the obstruction set is as follows. Note that the algorithm is always working with an obstruction set which is too small and so has a hypothesis class which is too big. Hence the equivalence queries always deliver a negative counterexample. In addition, when added to the set `Obstruct`, it is a minimal member of the complement of $\mathcal{G}$ in the minor relation. Hence, `Obstruct` is always a subset of the complete obstruction set.

```
Obstruct := {};
While Not EquivalenceQuery(Obstruct, CounterExample) Do Begin
  /* CounterExample is the negative counterexample returned by
     the equivalence query */
  Repeat
    Minors := OneStepMinors(CounterExample);
    /* Minors is the set of minors obtained from CounterExample
       by one step e.g. one edge deletion, etc.     */
    Smaller := False;
    Repeat
       Select G from Minors;
       If MembershipQuery( G ) Then
            /* i.e. G is a positive example  */
            Minors := Minors - { G }
       Else Begin
            CounterExample := G;
            Smaller := True;
       End;
    Until Smaller Or (Minors = {});
  Until Not Smaller;
  Obstruct := Obstruct + { CounterExample };
End;
Return ( Obstruct );
```

Observe that if the graph $G \preceq H$ with $G \neq H \in$ `Obstruct`, we must necessarily have $G \in \mathcal{G}$, since $G \notin \mathcal{G}$ would imply that H was not minimal in the complement of $\mathcal{G}$. We will prove by induction that at every stage the set formed by the set `Obstruct`$\cup\{$`CounterExample`$\}$ form an antichain in the relation $\preceq$, which represents a class $\mathcal{G}($`Obstruct`$\cup\{$`CounterExample`$\})$ which contains the class $\mathcal{G}$. This is certainly true at the start of the algorithm, since $\mathcal{G}(\emptyset)$ is the class of all graphs. Assume that it is true at some later iteration of the loop. If the equivalence query succeeds, then the algorithm has successfully identified the obstruction set. If it fails the counterexample `CounterExample` generated must be a negative example, since we have $\mathcal{G}($`Obstruct`$) \supseteq \mathcal{G}$. Hence also `Obstruct` $\cup$ $\{$`CounterExample`$\}$ form an antichain in the relation $\preceq$, since $G \succeq H \in \texttt{Obstruct}$ implies $\texttt{CounterExample} \notin \mathcal{G}(\texttt{Obstruct})$, while $G \preceq H \in \texttt{Obstruct}$ implies $G \in \mathcal{G}$ by the observation above. The other stage at which `CounterExample` is updated is in the inner loop after a membership query fails on a graph G which is a one step minor of `CounterExample`. Since $\texttt{Obstruct} \cup \{\texttt{CounterExample}\}$ formed an antichain and $G \preceq \texttt{CounterExample}$, no element of `Obstruct` is a minor of G. Similarly, $G \preceq H \in \texttt{Obstruct}$ implies $G \in \mathcal{G}$ by the observation. It follows that the set $\texttt{Obstruct} \cup \{G\}$ also forms an antichain. But as $G \notin \mathcal{G}$ we also have $\mathcal{G}(\texttt{Obstruct} \cup \{G\}) \supseteq \mathcal{G}$ as required.

In order to estimate the complexity of the algorithm, note that each iteration of the outer loop extends the set `Obstruct` by one element, and so the number of iterations is equal to the number of elements in the obstruction set. For each new counterexample, each execution of the the inner loop reduces its size by either one edge, or at least one vertex. Hence the number of iterations of that loop is linear in the size of the counterexample. Finally, the innermost loop is executed a number of times that is at most equal to the size of the counterexample. Hence, the time complexity of the two inner loops is at most quadratic in the size of the counterexample. By being more selective about the one step minors considered and taking into account that if an edge/vertex cannot be deleted at one step the same edge/vertex cannot be deleted at the next, these two inner loops can be made to run in time linear in the size of the counterexample. ∎

Corollary 7 *The class of all classes of graphs that are closed under taking minors can be learned using equivalence and membership queries.*

Proof: By the theorem, we can use the equivalence and membership queries to construct the obstruction set of the class, which is thus uniquely identified. ∎

Note that the obstruction set can then be used to determine membership of a given graph G in the class, in time that is $O(|G|^3)$, by Corollary 4. Further by Proposition 5, the test for membership can be made in $O(|G|)$ time, if the class does not contain all planar graphs.

4 Conclusions

Our results show that the learning protocol of membership and equivalence queries can be used to construct the obstruction set of a class of graphs closed under taking minors using time polynomial in the size of the obstruction set and the length of the longest counterexample. There are no known efficient algorithms for constructing the obstruction set even for simply defined classes. The algorithm can therefore also throw light on this open problem.

Once constructed the obstruction set makes it possible to check membership in the class in time polynomial in the size of the input graph. Indeed, if the class does not contain all planar graphs, the membership test can be performed in linear time using results concerning graph treewidth. Hence, if we have an efficient algorithm for testing membership in the class, we could construct a probably approximately correct obstruction set by simulating equivalence queries using a number of randomly chosen graphs. If they were all found to be correctly classified by the obstruction set so far obtained, we would be confident that future examples would be correctly classified by the given construction set. If on the other hand a graph was obtained that was incorrectly classified, we could use it in our algorithm as the graph returned by the equivalence query. This simulation technique is a standard method for reducing a pac learning problem to one using equivalence queries.

The results presented here can be generalised to any other partial order $\preceq$ on graphs as long as the defined class contains a finite number of obstructions, the number of immediate predecessors in the ordering is polynomial in the size of the graph and as long as we can check for any fixed H and an arbitrary G whether $H \preceq G$ in time polynomial in $|G|$. Examples of such partial orders are the immersion order and more generally the so-called Robertson and Seymour posets [8].

Note also that we do not need to restrict ourselves to graphs. Consider the set of all binary n-bit strings with the ordering being the monotonicity ordering. A string s can be characterised by monomial $m(s)$ containing the 1 coordinates. The set of strings t satisfying $s \preceq t$ in the ordering are precisely those satisfying the monomial $m(s)$. Hence, the set of strings determined by an obstruction set S are those satisfying the negation of the disjunctive normal form

$$\bigvee_{s \in S} m(s).$$

In this case Theorem 6 is the well-known result that monotone DNF formulae can be learned using membership and equivalence queries in time that is polynomial in the number of terms in the formula and the parameter n (the size of all counterexamples) [2].

One might be tempted to conjecture that all classes that can be learned by equivalence and membership queries satisfy this poset property. The poset structure would be defined in terms of the expressive power of the class $\mathcal{H}$ of functions considered. Hence, for inputs x and y, we would define

$$x \preceq y \Leftrightarrow f(x) = 0 \Rightarrow f(y) = 0,$$

for all $f \in \mathcal{H}$. A counterexample to this conjecture is provided by the languages recognised by DFA's, since for any pair of strings x, y there exist DFA's A and B, such that $A(x) = 0 = B(y)$ and $A(y) = 1 = B(x)$. Hence, x and y are not in the relation $\preceq$, which is thus the identity relation. This means that every string not in a language has to be specified which in turn implies that the obstruction set is not always finite.

Some other classes of functions satisfying the poset structure have been proved to be learnable only with membership queries, as read-once monotone formulas [6] or 2-monotonic positive boolean functions [9]. However, in our case equivalence queries seem essential for our algorithm as well as membership queries.

As we pointed out before, the representation class chosen is crucial for the learnability since the Learner has to find the correct representation. Taking as examples the regular languages, we know they are learnable if they are represented as Deterministic Finite Automata [1]. However, the same result probably does not hold if we represent them either as a Non Deterministic Automata or as a Regular Expression.

5 Acknowledgments

The authors would like to express their appreciation to the organizers of the ALCOM summer school on complexity, last September in Barcelona. It was during this meeting that the main ideas of this paper were born. We would also like to thank Ricard Gavaldà for helpful comments on an earlier version of this paper.

References

[1] D. Angluin. Learning regular sets from queries and counterexamples, *Information and Computation*, 75 (1987) 87–106.

[2] D. Angluin. Queries and Concept Learning, *Machine Learning*, 2 (1988) 319–342.

[3] D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5 (1990) 121-150.

[4] D. Angluin. Computational Learning Theory: Survey and Selected Bibliography. *Proceedings of the 24th ACM STOC*, (1992) 351-369.

[5] D. Angluin. Learning with Queries. *Computational Learning and Cognition.* Proceedings of the Third NEC Research Symposium, Princenton, ed. SIAM, (1993).

[6] D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *Tech. rep. University of California at Berkeley*, Report No. 89/528 (1989). J.ACM, to appear.

[7] H.L. Bodlaender. A Tourist Guide through Treewidth, *Acta Cybernetica*, 11 (1993) 1–21.

[8] M.R. Fellows and M.A. Langston. Exploiting RS-posets: Constructive algorithms from nonconstructive tools. Preprint, Feb. 1989.

[9] K. Makino and T. Ibaraki. A Fast and Simple Algorithm for Identifying 2-Monotonic Positive Boolean Functions *Technical Report of IEICE, COMP94-46* (1994) 11-20.

[10] N. Robertson and P.D. Seymour. Graph minors. I. Excluding a forest. *J. Comb. Theory Series B*, 35 (1983) 39–61.

[11] N. Robertson and P.D. Seymour. Graph minors. III. Planar tree-width. *J. Comb. Theory Series B*, 36 (1984) 49–64.

[12] N. Robertson and P.D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7 (1986) 309–322.

[13] N. Robertson and P.D. Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory Series B*, 41 (1986) 92–114.

[14] N. Robertson and P.D. Seymour. Graph minors. IV. Tree-width and well-quasi-ordering. *J. Comb. Theory Series B*, 48 (1990) 227–254.

[15] O. Watanabe. A formal study of learning via queries. *Lecture Notes in Computer Science 443: Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, M.S.Paterson, ed., Springer-Verlag, (1990) 139-152.

[16] O. Watanabe. A framework for polynomial time query learnability *Math. Systems Theory*, 27 (1994) 211–229.

[17] O. Watanabe and R. Gavaldà. Structural analysis of polynomial time query learnability. *Math. Systems Theory*, 27 (1994) 231-256.