To appear in *Proc. EuroCOLT '97*, which will be held in March 1997 in Jerusalem.

Learning from Incomplete Boundary Queries Using Split Graphs and Hypergraphs (Extended Abstract)

Robert H. Sloan[1]* and György Turán[2]**

[1] Dept. of EE & Computer Science, University of Illinois at Chicago
851 S. Morgan St. Rm 1120, Chicago, IL 60607-7053, USA
[2] Dept. of Mathematics, Stat., & Computer Science, University of Illinois at Chicago,
Research Group on Artificial Intelligence, Hungarian Academy of Sciences

Abstract. We consider learnability with membership queries in the presence of incomplete information. In the incomplete boundary query model introduced by Blum et al. [7], it is assumed that membership queries on instances near the boundary of the target concept may receive a "don't know" answer.

We show that zero–one threshold functions are efficiently learnable in this model. The learning algorithm uses split graphs when the boundary region has radius 1, and their generalization to split hypergraphs (for which we give a split-finding algorithm) when the boundary region has constant radius greater than 1. We use a notion of indistinguishability of concepts that is appropriate for this model.

1 Introduction

Learning in the presence of noise is one of the major issues that must be dealt with in order to design practical learning algorithms. Finding noise models that are both tractable and realistic is a nontrivial question in itself. For *PAC* learning, some of the the models considered are random noise affecting only the labels of examples, arbitrary noise affecting labels, arbitrary malicious noise, and, for instance space $\{0,1\}^n$, various models involving random noise affecting the instance bits [4, 11, 21, 23].

There are several recent approaches dealing with noise in query-based models of learning, in particular, for learning with membership and equivalence queries. Sakakibara [20] proposed a model where membership queries receive the wrong answer at random, but the noise is not persistent for any one instance, so repeated queries can overcome that noise. Angluin and Slonim [5] introduced a model where a random subset of the membership queries asked persistently receive the answer, "(I) don't know." Queries that may receive that response are referred to as *incomplete*. We previously studied the case where an adversary

* Partially supported by NSF grant CCR-9314258. Email: `sloan@eecs.uic.edu`.
** Partially supported by NSF grant CCR-9208170, OTKA grant T-14228, and Phare TDQM grant 9305-02/1022 (ILP2/HUN). Email: `U11557@uicvm.uic.edu`.

picks the subset of queries that receive a "don't know" response, and Angluin and Kriķis studied a model where an adversary picks a subset of queries that receive the wrong response [2, 3, 22].

A practical motivation for studying noisy membership queries comes from the experiments of Lang and Baum [15], which confirm the intuition that erroneous answers to membership queries are more frequent close to the boundary of the target concept. Blum et al. [7] formulated interesting models that try to capture this phenomenon by assuming that the answers to the membership queries are all correct *except in a neighborhood of a fixed radius* of the boundary of the target concept. (A point is close to the boundary of the target concept if its classification can change if a few of its components are flipped.) In the neighborhood of the boundary the answers to the membership queries are assumed to be either correct or "don't know" (the *incomplete boundary query* (IBQ) model), or they are assumed to be either correct or incorrect (the *unreliable boundary query* (UBQ) model). These models are extended to allow equivalence queries by requiring that the counterexamples be chosen from outside the boundary region. Efficient learning algorithms are required to be polynomial in the usual learning parameters. Thus, although the boundary of the target concept may be exponentially large, and so there may be exponentially many instances that receive a "don't know" answer, a learning algorithm is still required to identify the target concept after only polynomially many queries.

One of the main tasks in the area of learning from noisy data is to develop further learning algorithms for basic concept classes that may lead to general techniques for noise-tolerant learning. In this paper, we provide a step in this direction. We consider the incomplete boundary query model, which is the milder of the two models proposed in [7], and we discuss learning from membership queries alone (thus, we do not use equivalence queries).

Blum et al. [7] gave a learning algorithm for the intersection of two halfspaces in Euclidean space in the UBQ model. For the Boolean case, they considered learning subclasses of monotone DNF in a modified version of the UBQ model with only one-sided error. Notice that every learning algorithm in the UBQ model or the one-sided UBQ model can run in the IBQ model by changing every "don't know" answer it receives to 1. Thus some subclasses of monotone DNF are efficiently learnable in the IBQ model.

In this paper, we look at the learnability of *zero–one threshold functions* in the IBQ model. These are functions of the form $Th_k(\mathbf{y})$, where $\mathbf{y} = (y_1, \ldots, y_m)$ is a subset of the variable set $\{x_1, \ldots, x_n\}$. The variables in $(y_1, \ldots, y_m)$ are the *relevant variables* and k is the *threshold*. An instance $v \in \{0,1\}^n$ belongs to the concept $Th_k(\mathbf{y})$ if at least k of the variables $(y_1, \ldots, y_m)$ have value 1 in v.

Zero–one threshold functions are a well-studied class [13, 14, 16, 19]. In particular, Hegedűs showed that this class can be learned with $O(n)$ membership queries [14]. In fact, one can even learn the larger class of *read-once compositions* of zero–one threshold functions (i.e., zero–one threshold formulas) with polynomially many membership queries [8]. On the other hand, *weighted monotone threshold functions*, that is, monotone halfspaces in $\{0,1\}^n$, *cannot* be learned with polynomially many membership queries [6].

The (noise-free) membership query algorithm for learning zero–one threshold functions [14] is as follows. Starting with the all-ones vector 1^n, turn components to 0 as long as this still gives a positive example of the target concept. The variables that are set to 1 in the final vector are all relevant, and their number is equal to the threshold of the target concept. Turning one more 1 to 0 we get a negative example. The remaining relevant variables are those 0 components of this negative example, which, when turned to 1, give a positive example.

This algorithm fails if answers to boundary queries may be incomplete. In fact, all the essential information obtained is in the boundary region of the target concept.

In this paper we show that zero–one threshold functions are *learnable with polynomially many incomplete boundary queries.* We give a detailed proof for the case when the radius of the boundary region is 1, and we outline the proof for the general case of radius r.

The learning algorithm for zero–one threshold functions is based on the observation that, given a positive example near the boundary of the target concept, switching a sufficient number of relevant variables from 1 to 0 will "break on through to the other side," and thus a query to this modified instance will receive a negative (i.e., 0) answer from the oracle. On the other hand, switching a set of irrelevant variables from 1 to 0 will lead to a positive example. Hence the oracle must give a positive (i.e., 1) or "don't know" answer to these queries. This gives a dichotomy that can be used to identify a set of candidates for the minimal point found in the first stage of the noise-free algorithm. A similar idea can be used to simulate the second stage of the algorithm.

The dichotomy can be formulated conveniently in terms of *split graphs* for the case of radius 1, and in terms of *split hypergraphs* in the general case. Split graphs were defined by Földes and Hammer [9] (see also [12, 17].). Split graphs have nice characterizations and efficient algorithms. The general case of split hypergraphs does not appear to have been studied before. We give an algorithm to determine whether an r-uniform hypergraph is a split hypergraph, and if so, to output a split, in polynomial time. This is used in showing that our learning algorithms require only a polynomial amount of computation.

Another component of our learning algorithm is a characterization of a notion of *indistinguishability* of two concepts, which is used to filter out those candidate hypotheses that are not acceptable as outputs of the learning algorithm.

It is hoped that the method used in the learning algorithm will be useful for learning other classes with incomplete boundary queries. In the last section of this paper, we mention some of the numerous open problems suggested by this work.

2 Definitions

We follow the standard model of learning from membership queries. The goal of the learner is to infer an unknown *target concept* from some given *concept class*

$\mathcal{C}$ over a given *instance space* or *domain*. We will view concepts interchangeably as subsets of the instance space and as 0–1 functions on the instance space.

For the standard noise-free model, *proper exact identification* is required. This means that the learner must output precisely the correct target concept, and that the output must be in a specified syntactic form.

Fix a target concept C. An ordinary or *complete membership query* on instance v, denoted $\mathrm{MQ}(v)$, returns $C(v)$. In this paper we study the *incomplete boundary membership oracle*, proposed by Blum et al. [7]. For the instance space $\{0,1\}^n$, the *distance to the boundary* of instance v is the Hamming distance to the nearest instance w such that $C(w) \neq C(v)$. The *boundary region of radius r of C* is the set of all instances whose distance to the boundary is at most r. The incomplete boundary oracle returns either the correct classification of the instance, $C(v)$, or "don't know," which we will denote by $*$. The response $*$ can be given only for instances in the boundary region of radius r. The radius is a fixed, known constant.

Queries to this oracle about instance v will be denoted $\mathrm{IBQ}(v)$. In what follows, we often use expressions such as "v has IBQ 1" for $\mathrm{IBQ}(v) = 1$. A *positive instance* means any instance v in the target concept regardless of whether $\mathrm{IBQ}(v) = 1$ or $\mathrm{IBQ}(v) = *$, and similarly for negative instances.

The requirement for successful learning in this noise model is that we must output a hypothesis that agrees with the target concept on all instances that are not in the boundary region of the target concept [7]. Moreover, the learner is allowed only a number of queries bounded by a polynomial in n, regardless of the frequency of $*$ responses. In general, there will be exponentially many instances in the boundary region even for constant r.

Let us briefly compare this model to the *limited membership query model* [3, 22]. On the one hand, in the limited membership query model, the oracle is allowed to respond $*$ on any instance it chooses, not only on boundary instances. On the other hand, in that model the learner is allowed a number of queries polynomial in both *n and the number of $*$ responses received*. In particular, our algorithm for learning monotone DNF from limited membership plus equivalence queries could receive an exponential number of $*$ responses (in the "Up" phase of that algorithm [3, 22]) even if the $*$ responses are restricted to a boundary region of radius 1. Thus the two models appear to be incomparable.

2.1 Terminology for $\{0,1\}^n$

As do others (e.g., [5, 10]), we will view $\{0,1\}^n$ as a partially ordered set. The top element is the vector 1^n and the bottom element is 0^n. The *descendants* of an instance v are all instances w such that $w \leq v$, where $x \leq y$ if each bit of x is less than or equal to the corresponding bit of y. The *ancestors* of an element v are all elements w such that $w \geq v$. The *children* of v are all descendants of v with Hamming distance 1 from v; the *parents* of v are all ancestors of v with Hamming distance 1 from v. The norm $|v|$ of v is the number of v's components set to 1.

A *minterm* of a function f is a set S of variables such that for any assignment v that sets all the variables in S to 1, it holds that $f(v) = 1$, but this is not true for any subset of S. We will sloppily refer to both S and vectors from $\{0,1\}^n$ with 1's in exactly those positions corresponding to S as minterms. Note that for a zero–one threshold function, the norm of a minterm is exactly the threshold.

3 Split Graphs and Hypergraphs

An undirected graph $G = (V, E)$ is a *split graph* if V has an ordered partition (V_1, V_2) such that G induces a clique on V_1 and an independent set on V_2. We note here two useful facts about split graphs. First, there is a linear time algorithm to determine whether a graph is a split graph, and output a split if so (see [17]). Second, any split graph has at most $|V| + 1$ distinct splits.

An *r-uniform hypergraph* H has the form $H = (V, E)$, where V is a set and E is a set of r-element subsets of V. An r-uniform hypergraph is a *split hypergraph* if V has an ordered partition (V_1, V_2) such that E contains all r-element subsets of V_1, and E contains no r-element subset of V_2. In this case (V_1, V_2) is called a *split* of H.

Lemma 1. *For every fixed r, there is a polynomial-time algorithm that decides if an r-uniform hypergraph is a split hypergraph, and finds a split if it is.*

Proof Let H be a split hypergraph and let (V_1, V_2) be a split. Then for every other split (U_1, U_2) it must hold that $|V_1 \cap U_2| < r$ and $|V_2 \cap U_1| < r$. Thus U_1 can be obtained by deleting fewer than r elements from V_1 and adding fewer than r elements to V_1. Hence the number of splits is $O(n^{2r-2})$, where $|V| = n$.

Thus *all* splits of a hypergraph can be found in polynomial time by divide and conquer. After finding all splits in the hypergraphs induced by H on two arbitrarily chosen halves of V, we combine the splits in all possible ways. This gives $O(n^{4r})$ partitions of V. For each partition, we check whether it forms a split of H. Checking whether a partition is a split requires the examination of $O(n^r)$ subsets. Thus the overhead is polynomial. As the restriction of a split to a subset is a split of the hypergraph induced on this subset, the algorithm finds all splits of H. □

4 Algorithm for Boundary Radius 1

In this section we describe our learning algorithm for the case of boundary radius 1. This is extended to the general case of radius r in the next section.

Notice that for boundary radius 1, monotone conjunctions are trivially learnable: The hypothesis "always false" meets the requirements of the IBQ model. Every true instance of a conjunction is within Hamming distance 1 of a false instance: Simply turn off one bit needed in the conjunction. Thus the hypothesis "always false" errs only on instances in the boundary region. Similarly, for boundary radius 1, monotone disjunctions are trivially learned by the hypothesis "always true." These observations will be used below to handle some special cases in the beginning of the proof.

Theorem 2. *Zero–one threshold functions are learnable from $O(n^3)$ incomplete boundary queries with boundary radius 1.*

The algorithm has three main stages. First we find a small ($O(n)$) set of candidates that is guaranteed to contain a minterm for the target function. Next, for each candidate minterm, we find a small (again $O(n)$) set of groups of variables such that, if the candidate minterm is correct, then one of those groups of variables must be the remaining relevant variables of the target function. Sometimes in this step we instead discover that our candidate minterm is in fact not a minterm. This gives us a pool of at most $O(n^2)$ hypotheses, which must contain the target function. The third stage of our algorithm prunes this pool to a set of hypotheses that are all acceptable outputs for our algorithm.

We begin with the query IBQ(1^n). If the response is 0 or $*$, then our algorithm responds "always false," and terminates. (For response 0, the target must be "always false." The "always false" function has no boundary instances, so for response $*$, the target must be a monotone conjunction.) Otherwise, the search for a minterm begins from the instance 1^n.

4.1 Finding a Minterm

We now show how to find a set of candidate minterms that must contain a correct minterm.

The first step is the appropriate noisy variation of the "reduce" algorithm that is common to many algorithms using membership queries to learn some form of monotone DNF (e.g., [1]). Given that 1^n has IBQ 1, we set v to 1^n. Now, so long as v has some child to which a membership query responds 1, we replace v by that child and repeat. Thus v will eventually be an instance with IBQ 1 all of whose children are classified either 0 or $*$.

For the special case $|v| \leq 1$, we output the hypothesis "always true" and terminate the entire algorithm. When $v = 0^n$, "always true" is obviously the correct hypothesis. If $|v| = 1$, then the target function must be a monotone disjunction. (Since v's child, 0^n, was classified as either 0 or $*$, the target cannot be the "always true" function. That function has no boundary instances.)

In general, when $|v| \geq 2$, there will be three cases for finding candidate minterms, depending on the children of v.

Case I: If all of v's children are classified 0, then v is definitely a minterm.

In general, however, the final value of v from the "reduce" procedure may not be a minterm—we know only that v must contain a minterm. For example, let $n = 15$, and assume that $v = 1^8 0^7$, and all of v's children are classified $*$. It might be that a minterm is $x_1 \cdots x_8$, but the IBQ answers are also consistent with a minterm being $x_1 \cdots x_3$, with $x_4, \ldots, x_8$ all being irrelevant variables.

Case II: v has at least one child classified 0. Let k be the threshold of the target function. Now v must contain exactly k relevant 1's and an unknown number of irrelevant 1's. We determine (approximately) which 1's are relevant as follows.

We construct a complete undirected graph, G_1, with labeled edges. The vertices of G_1 are the variables that are assigned to 1 in v. Recall that $|v| \geq 2$. Every edge will be labeled either 0 or $*$. Each edge corresponds to a pair of variables x_i and x_j. This pair of variables, in turn, corresponds to the grandchild of v obtained by turning off x_i and x_j. The label of this edge is the answer received from an IBQ on that grandchild. If any such grandchild has IBQ 1, then we resume the reduce process with that grandchild. Thus in what follows we assume that this does not happen.

When discussing split graphs in what follows, it is more convenient to consider complete graphs with edges having two different kinds of labels (e.g., 0 or $*$), instead of considering cliques and independent sets.

Lemma 3. *Graph G_1 is a split graph.*

Proof sketch. Any grandchild that had two relevant variables turned off must have classification 0, since it has distance 2 from the boundary. Also, any grandchild that had two irrelevant variables turned off must have classification $*$ (or 1 but then we would have gone back to reducing), since it is a positive instance. □

We find all possible splits of G_1. This can be done, for example, using Lemma 1. (For the $r = 1$ case, there are more efficient algorithms.) For each split, the set of variables in the half of G_1 with edges labeled 0 is a candidate minterm.

Case III: all of v's children are classified $*$. Because v is a positive example, v contains at least k relevant 1's. It is possible that v contains exactly k relevant 1's, so we first do the same thing as we did in Case II. If the resulting graph is *not* a split graph, then we know that v contains more than k relevant 1's. If the graph is a split graph, then we obtain some candidate(s) for being a minterm.

Regardless of whether that graph was a split graph, it is possible that v contains more than k relevant 1's. Then v must not contain any irrelevant 1's, since then at least one of its children would be classified 1 (instead of $*$). Also, v cannot have more than $k+1$ relevant 1's, because in that case v's children would not be in the boundary region. Thus v must contain precisely $k+1$ relevant 1's, no irrelevant 1's, and all of v's children are minterms. Therefore, we also add as a candidate minterm one arbitrarily selected child of v.

Lemma 4. *The procedure described above outputs a set of at most $n+2$ candidate minterms, and that set is guaranteed to contain at least one minterm of the target function.*

Proof sketch. The existence of the minterm was discussed in the development of the algorithm.

The number $n + 2$ is derived as follows. As noted previously, the largest possible number of splits of a graph with $|V|$ vertices is $|V| + 1$. The largest possible number of 1's in v is n, so the split graph construction could lead to at most $n+1$ candidates for the minterm. One other candidate can be added for a v with all $*$ children. □

4.2 Finding the Other Relevant Variables

If our candidate minterm is $x_1 \cdots x_n$, then the only possible target function is the conjunction of all variables.

Now we assume that we have a candidate minterm v with $k < n$ variables. We need to find the remaining relevant variables in our target concept. Obtain w from v by changing exactly one of v's 1's to a 0. We then query every instance formed from w by changing to 1 a pair of variables that are both assigned 0 by v. That is, we query all grandparents of w that are not parents of v.

For any of those queries that returns 0, we mark both of the added variables as irrelevant, and discard them. We then form a graph G_2^v on the remaining variables assigned to 0 by v. The definition of G_2^v is analogous to that of graph G_1 in the previous subsection, with the queries now being of grandparents instead of grandchildren. So the edges of G_2^v are all labeled either 1 or $*$. Once again, if v is a minterm, then G_2^v must be a split graph: Any variable setting with $k-1$ variables from v plus two additional relevant variables must be classified 1, whereas any setting with only $k-1$ relevant variables set to 1 must be classified $*$ (or 0, but we discarded those variables).

If G_2^v is not a split graph, then we know that v cannot have been a minterm. Otherwise, the clique labeled 1 from each possible split gives us one possibility for the remaining relevant variables in the target function. We summarize this with a lemma.

Lemma 5. *The procedure described above, on input a correct minterm v, outputs a set of at most $n - |v| + 1$ sets of variables, and one of those sets is exactly the additional relevant variables of the target function.* □

4.3 Pruning the Set of Candidates

After running the algorithms of the previous two subsections, we obtain a set of of $O(n^2)$ hypotheses that contains the correct hypothesis. We must now prune this set so that it contains only acceptable hypotheses.

We call two concepts *r-indistinguishable* if their symmetric difference is in the intersection of their boundary regions of radius r, and *r-distinguishable* otherwise. An *r-distinguishing instance* for a pair of r-distinguishable concepts is an instance from their symmetric difference that is *not* in the intersection of their boundary regions of radius r. (We omit the r when it is clear from context.)

Lemma 6. *Given two r-distinguishable concepts f_1 and f_2, and an r-distinguishing instance v, the result of* IBQ(v) *will be inconsistent with at least one of f_1 or f_2.*

Proof sketch. This follows immediately from the definitions, noting that a $*$ answer is inconsistent with at least one of f_1 or f_2. □

Lemma 6 tells us how to use an r-distinguishing instance to prune the set of candidate hypotheses; the following lemma tells us how to find such an r-distinguishing instance.

Lemma 7. *If two zero–one threshold functions f_1 and f_2, with thresholds k_1 and k_2, respectively, are r-distinguishable, then at least one of the following instances is an r-distinguishing instance.*

1. *Setting k_1 of f_1's variables to 1, using as few of f_2's relevant variables as possible.*
2. *Setting $k_1 + r$ of f_1's variables to 1, using as few of f_2's relevant variables as possible.*
3. *One of the two symmetric possibilities for f_2.*

Proof sketch. Since f_1 and f_2 are distinguishable, there must be an instance q in their symmetric difference that is not in the boundary region of at least one of f_1 or f_2. We consider the case where q is positive for f_1 and negative for f_2. The opposite case is symmetric.

We begin with an instance q_1 formed according to the rules given in Case 1 of the statement of the theorem. Since there are instances that are positive for f_1 and negative for f_2, and q_1 is a positive instance of f_1 with a minimum number of f_2's relevant variables set to 1, we know that $f_2(q_1) = 0$. If q_1 is not in f_2's boundary region, then $q = q_1$ and we are done.

Otherwise, no positive boundary instance for instance for f_1 is a distinguishing instance. Therefore, the distinguishing instance q must be positive nonboundary for f_1 and negative for f_2. It suffices to construct q to have the minimum number of f_2's relevant variables set to 1 among all positive nonboundary instances of f_1. Case 2 of the statement of the theorem does precisely that. (An example for $r = 1$ when Case 2 is necessary is given by $f_1 = Th_2(w, x, y, z)$, $f_2 = Th_1(w)$, $q_1 = 0110$, and $q = 0111$.) □

Conclusion of the proof of Theorem 2. Lemmas 6 and 7 give us an algorithm to prune the set of hypotheses so that every remaining pair of hypotheses is 1-indistinguishable. We simply query distinguishing instances for pairs of distinguishable hypotheses, discarding those hypotheses that are inconsistent with the answer. (Note that we can compute whether a pair of hypotheses is distinguishable without making any queries.) The pruning will not delete the target function from our set of candidates. Therefore every hypothesis in the final pruned set will be 1-indistinguishable from the target concept, and so is an acceptable output for the learning algorithm.

We note that Theorem 8, which follows, tells us that there can be at most $n + 1$ hypotheses in the pool when the algorithm terminates.

A crude analysis gives a total query complexity of $O(n^3)$. Notice that all the computation is polynomial. □

Lemma 7 can also be used to tell precisely when two zero–one threshold functions are r-indistinguishable. We give this description for the case $r = 1$.

Theorem 8. *Two threshold functions are 1-indistinguishable if and only if they fall into one of the following categories.*

1. *They have the same set of relevant variables, and their thresholds differ by 1.*
2. *One function has one extra relevant variable, and the two functions have either the same threshold or the function with an extra relevant variable has a threshold 1 greater than the other function.*
3. *Each function contains one relevant variable that the other does not, and the two functions have the same threshold.*
4. *One function has two extra relevant variables, and that function has a threshold 1 greater than the other function.*

Proof sketch. It is straightforward to verify that in each of those four cases the two functions meet the definition of indistinguishability. Arguing that those are the only possibilities requires a somewhat lengthy case-by-case discussion, which we omit from this extended abstract. □

5 Arbitrary Constant Radius

Now we generalize Theorem 2 to the case of larger boundary radii.

Theorem 9. *Zero–one threshold functions are learnable from polynomially many incomplete boundary queries for any constant boundary radius r.*

Proof sketch. We will just briefly give some of the key differences between the $r = 1$ case of the previous section and the arbitrary constant r case. Throughout, let k be the threshold of the target function f.

The search for a minterm proceeds as before to some instance v with IBQ 1 none of whose children have IBQ 1. There are the same three cases as before: all of v's children have IBQ 0, some of v's children have IBQ 0 and some have IBQ $*$, and all of v's children have IBQ $*$. Again, in the first case, we know v is a minterm.

In the second case, we again know that v has exactly k relevant 1's. If v has exactly k relevant 1's, then the $(r + 1)$-uniform hypergraph whose vertices are the variables that are 1 in v formed by considering all $(r + 1)$-level descendants of v will be a split hypergraph: If we turn off $r + 1$ of v's relevant 1's, then we obtain a nonboundary negative instance, and if we turn off $r + 1$ irrelevant 1's, then we obtain a positive instance.

If none of v's children has IBQ 0 (i.e., all have IBQ $*$), then we must do more. In this case, v could contain as many as $k + r$ relevant 1's. Thus we run the preceding construction from every descendant of v that has IBQ $*$ and is between 1 and r levels below v. This ensures that the construction is run from some instance with exactly k relevant 1's. Since r is constant, we run the construction only polynomial in n many times.

Now we assume that we have a candidate minterm v that has k relevant 1's. We turn off exactly one of v's 1's, and build the hypergraph with labeled edges formed by IBQs to assignments completed by turning on all $(r+1)$-tuples of variables assigned to 0 by v. As with the $r = 1$ case, an answer of 0 means that all $r+1$ variables are irrelevant, and we can therefore discard all those variables. Among the remaining variables, the relevant variables form a clique labeled 1, and the irrelevant variables form a clique labeled $*$.

Pruning the set of candidate hypotheses is very similar to the $r = 1$ case, because Lemmas 6 and 7 are not restricted to the case $r = 1$. $\square$

6 Some Open Problems

It would be interesting to give efficient incomplete boundary query algorithms for other concept classes, which would probably require adding equivalence queries to the model. Can the results of this paper be extended to read-once disjunctions of zero–one threshold functions, or, more generally, to the class of zero–one threshold formulas [8], and to the nonmonotone versions of these classes?

Another problem is to extend the result of this paper to unreliable boundary queries, that is, to the case of lies instead of "don't know"s.

It would also be of interest to look at the relationship between the incomplete boundary query model and the incomplete membership query model [22], and to prove lower bounds for the number of incomplete boundary queries. Can zero–one threshold functions be learned with less than $\Omega(n^3)$ incomplete boundary queries?

There are some combinatorial questions related to split hypergraphs. Split graphs have a forbidden induced subgraph characterization: Split graphs are exactly those graphs which do not contain cycles of length 4 or 5, or two independent edges as induced subgraphs [9]. Peled [18] asked if split hypergraphs have a forbidden induced subgraph characterization. Another question concerns the complexity of finding a split in a split hypergraph. The algorithm described in Section 3 is far from optimal for $r = 1$. Is there a more efficient algorithm for $r > 1$?

In the discussion of split hypergraphs it was assumed that the hypergraphs are given, as usual, by a list of their edges. If the radius r is constant, this representation is polynomial in n. On the other hand, in our setting the hypergraphs are given *implicitly* in the form of the oracle answering the queries. Thus it would be interesting to know if a split can be found using this oracle model, even when r grows with n. This would give an efficient incomplete boundary query algorithm for learning zero–one threshold functions even when the radius of the boundary region grows with n.

Acknowledgments

We would like to thank Uri Peled for helpful conversations about split graphs, and Tibor Hegedűs for valuable comments on a draft of this paper.

References

1. D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, Apr. 1988.
2. D. Angluin and M. Kriķis. Learning with malicious membership queries and exceptions. In *Proc. 7th Annu. ACM Workshop on Comput. Learning Theory*, pages 57–66. ACM Press, New York, NY, 1994.
3. D. Angluin, M. Kriķis, R. H. Sloan, and G. Turán. Malicious omissions and errors in answers to membership queries. *Machine Learning*. To appear.
4. D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.
5. D. Angluin and D. K. Slonim. Randomly fallible teachers: learning monotone DNF with an incomplete membership oracle. *Machine Learning*, 14(1):7–26, 1994.
6. M. Anthony, G. Brightwell, D. Cohen, and J. Shawe-Taylor. On exact specification by examples. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 311–318. ACM Press, New York, NY, 1992.
7. A. Blum, P. Chalasani, S. A. Goldman, and D. K. Slonim. Learning with unreliable boundary queries. In *Proc. 8th Annu. Conf. on Comput. Learning Theory*, pages 98–107. ACM Press, New York, NY, 1995.
8. N. Bshouty, T. Hancock, L. Hellerstein, and M. Karpinski. An algorithm to learn read-once threshold formulas, and transformations between learning models. *Computational Complexity*, 4:37–61, 1994.
9. S. Földes and P. L. Hammer. Split graphs. *Congressus Numerantium*, 19:311–315, 1977.
10. S. A. Goldman and H. D. Mathias. Learning k-term DNF formulas with an incomplete membership oracle. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 77–84. ACM Press, New York, NY, 1992.
11. S. A. Goldman and R. H. Sloan. Can PAC learning algorithms tolerate random attribute noise? *Algorithmica*, 14:70–84, 1995.
12. M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Computer Science and Applied Mathematics. Academic Press, New York, 1980.
13. Q. P. Gu and A. Maruoka. Learning monotone boolean functions by uniformly distributed examples. *SIAM J. Comput.*, 21:587–599, 1992.
14. T. Hegedűs. On training simple neural networks and small-weight neurons. In *Computational Learning Theory: Eurocolt '93*, volume New Series Number 53 of *The Institute of Mathematics and its Applications Conference Series*, pages 69–82, Oxford, 1994. Oxford University Press.
15. K. J. Lang and E. B. Baum. Query learning can work poorly when a human oracle is used. In *International Joint Conference on Neural Networks*, Beijing, 1992.
16. N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
17. N. V. R. Mahadev and U. N. Peled. *Threshold Graphs and Related Topics*, volume 56 of *Annals of Discrete Mathematics*. Elsevier Science B.V., Amsterdam, The Netherlands, 1995.

18. U. Peled. Personal Communication.
19. L. Pitt and L. Valiant. Computational limitations on learning from examples. *J. ACM*, 35:965–984, 1988.
20. Y. Sakakibara. On learning from queries and counterexamples in the presence of noise. *Inform. Proc. Lett.*, 37:279–284, 1991.
21. R. H. Sloan. Four types of noise in data for PAC learning. *Inform. Proc. Lett.*, 54:157–162, 1995.
22. R. H. Sloan and G. Turán. Learning with queries but incomplete information. In *Proc. 7th Annu. ACM Workshop on Comput. Learning Theory*, pages 237–245. ACM Press, New York, NY, 1994.
23. L. G. Valiant. Learning disjunctions of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence, vol. 1*, pages 560–566, Los Angeles, California, 1985. International Joint Committee for Artificial Intelligence.