Open problem: learning a function of r relevant variables

Avrim Blum*

Carnegie Mellon University

This problem has been around for a while but is one of my favorites. I will state it here in three forms, discuss a number of known results (some easy and some more intricate), and finally end with small financial incentives for various kinds of partial progress. This problem appears in various guises in [BFKL93,Blu94,MOS03]. To begin we need the following standard definition: a boolean function f over $\{0,1\}^n$ has (at most) r relevant variables if there exist r indices $i_1, \ldots, i_r$ such that $f(x) = g(x_{i_1}, \ldots, x_{i_r})$ for some boolean function g over $\{0,1\}^r$. In other words, the value of f is determined by only a subset of r of its n input variables. For instance, the function $f(x) = x_1\bar{x}_2 \vee x_2\bar{x}_5 \vee x_5\bar{x}_1$ has three relevant variables. The "class of boolean functions with r relevant variables" is the set of all such functions, over all possible g and sets $\{i_1, \ldots, i_r\}$. The problems are:

(a) Does there exist a polynomial time algorithm for learning the class of boolean functions that have $\lg(n)$ relevant variables, over the uniform distribution on $\{0,1\}^n$?

(b) Does there exist a polynomial time algorithm for learning the class of boolean functions that have $\lg\lg(n)$ relevant variables, over the uniform distribution on $\{0,1\}^n$? Notice that since there are only $2^{2^{\lg\lg n}}$ possible g's, we can assume the function g is known, and the only difficulty is determining which are the relevant variables.

(c) Does there exist an algorithm for learning the class of boolean functions that have r relevant variables, over the uniform distribution on $\{0,1\}^n$, in time "substantially" better than n^r?

Motivation: These problems are all a special case of the question of whether DNF or Decision Trees can be learned in polynomial time. In particular, any function of r relevant variables can be written as a decision tree of depth r (branching on a different variable at each level) or as a DNF formula with at most 2^r terms (one for each positive entry in the truth-table for g). These both have polynomial size for $r = O(\log n)$, and so progress on the relevant-variable problem is necessary for any positive answers to those questions. Furthermore, the relevant-variable problem is arguably more basic than the DNF or Decision-Tree questions, since the target class is defined semantically rather than syntactically. Lastly, these

* Supported in part by NSF grants CCR-0105488 and NSF-ITR CCR-0122581.

problems suggest specific "challenge" distributions on target functions that could be used to test heuristics (see below).

Current status: A recent and very nice result of Mossel, O'Donnell, and Servedio [MOS03], building on ideas of Kalai and Mansour [KM01], achieves roughly $O(n^{0.7r})$ for problem (c). In the other direction, there exists a *specific* function g for which (if the set of relevant variables is chosen at random), all current techniques appear to break down at $O(n^{r/3})$ [BFKL93]. In addition to the question of whether [MOS03] can be improved, a natural question is whether this specific case can be learned more efficiently, and whether one can construct "harder" functions g for which, say, beating $n^{r/2}$ appears hard. See below for rewards related to these statements.

Some observations:

1. If membership queries are allowed, then learning is easy. (This makes a good homework question). Given a positive and negative example, one can "walk" them together to identify one relevant bit, put that at the top of a decision tree, and then recursively learn each subtree. Or, one can apply the more general algorithms of Bshouty [Bsh93] or Jackson [Jac94].
2. If the target function is unbiased, then weak learning, strong learning, and exact identification are equivalent. (This also makes a good homework problem.) In particular, if A is a weak-learning algorithm, then one can identify a relevant variable by running A on data in which the first i bits of every example are replaced by new, truly random bits. If we do this for $i = 0, 1, \ldots, n$, then at some point A must fail to perform better than random guessing, and this will occur at one of the relevant variables. If f is a *biased* function, then for this to work we need to change the definition of "weak learning" to mean an algorithm that performs noticeably better than the underlying bias of the target function.
3. We can assume without loss of generality that the relevant variables are chosen at random (since the algorithm can always randomly permute the indices if it so chooses).
4. Here is a specific function g proposed in [BFKL93] as a candidate hard case. Split the relevant variables into two sets A and B. On input x, compute the Parity function over A, and the Majority function over B, and then XOR the two results together.[1] This class can be easily learned in time $O(n^{|A|})$ (by guessing the set A and reducing to majority) or in time $O(n^{|B|/2})$ (by guessing half of B and examining only the examples in which those bits are all 0, reducing to parity). The worst case is when $|A| = r/3, |B| = 2r/3$, yielding an algorithm that runs in time $O(n^{r/3})$, but no better algorithm is known.

[1] For instance, if $A = \{1, 2, 3\}$ and $B = \{4, 5, 6\}$ then the classification of the example 011101001010 would be positive, since the first three bits have an even number of ones (making their parity 0), and the next three bits have more ones than zeros (so the majority function is 1), and the XOR of those two quantities is 1.

Notice that this specific function g gives a samplable distribution on target functions f (pick a random subset of r variables, split into A and B, and feed it into g). Thus one can test proposed heuristics.

5. Problems (a) and (b) are not solvable by SQ algorithms [Kea98,BFJ+94]. This holds even for the specific g above. However, learning is easy for "most" functions g (e.g., if the truth table is picked at random). The difficult cases seem to be the functions g that are "similar to parity functions, but not exactly."
6. The theory of fixed-parameter tractability [DF95,DF99] has been used to analyze the complexity of problems as a function of the size of the solution. For example, suppose we want to determine if an instance of the set-cover problem has a cover of size r. If there are n sets total, then it is easy to do this in time $O(n^r)$, but the results of [DF95] suggest that achieving running time of $f(r)poly(n)$ for any function f (e.g., $f(r) = 2^{2^r}$) may be hard. This has immediate consequences for the "proper learning" problem, if we allow examples to be arbitrary. For instance, it implies that determining if the data is consistent with a conjunction of size r is likely to be hard even if $r = O(\log n)$. However it is unclear whether this theory can be used to show (or suggest) hardness for the prediction problem.

Monetary rewards:

$100: Improve the results of [MOS03] to $O(n^{0.666r})$.
$200: Improve the results of [MOS03] to $O(n^{0.499r})$.
$100: Find an algorithm to learn the class described in Observation (4) in time $O(n^{r/4})$.
$500: Find an algorithm to learn the class described in (4) in polynomial time, for $r = O(\log n)$.
$50: Find a function g as in (4) but for which achieving $O(n^{r/2})$ appears hard.
$1000: Give a positive solution to open problem (a) or (b).
$50+: Give a convincing argument why nobody will ever be able to solve the above $1000 problem. (Prize depends on how convincing.)

References

[BFJ+94] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. Weakly learning DNF and characterizing statistical query learning using fourier analysis. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 253–262, May 1994.

[BFKL93] A. Blum, M. Furst, M. Kearns, and D. Lipton. Cryptographic primitives based on hard learning problems. In D. Stinson, editor, *13th Annual International Cryptology Conference (CRYPTO)*. Springer, 1993. Lecture Notes in Computer Science No. 773.

[Blu94] Avrim Blum. Relevant examples and relevant features: Thoughts from computational learning theory. In *AAAI-94 Fall Symposium, Workshop on Relevance*, 1994.

[Bsh93] N. H. Bshouty. Exact learning via the monotone theory. In *Proceedings of the IEEE Symposium on Foundation of Computer Science*, pages 302–311, Palo Alto, CA., 1993.

[DF95] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness i: Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995.

[DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science of Springer-Verlag, 1999.

[Jac94] J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. In *Proceedings of the IEEE Symposium on Foundation of Computer Science*, 1994.

[Kea98] M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998.

[KM01] Adam Kalai and Yishay Mansour. Perosnal communication. 2001.

[MOS03] Elchanan Mossel, Ryan O'Donnell, and Rocco Servedio. Learning juntas. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, 2003.