The Minimal Disagreement Parity Problem as a Hard Satisfiability Problem

James M. Crawford
Computational Intelligence Research Laboratory
1269 University of Oregon
Eugene, OR 97403

Michael J. Kearns
Robert E. Schapire
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

February 1, 1994

1 Introduction

One common approach to generating hard instances for SAT algorithms is to encode other NP-complete problems into SAT. An obvious problem with this approach is that many NP-complete problems are trivially solvable in the average case [4]. One solution to this problem is to choose problem classes which are *randomly self-reducible*. As we show in section 3, randomly self-reducible problems are as hard in the average case as they are in the worst case.

The minimal disagreement parity (MPD) problem is a simple example of a randomly self-reducible problem.[1] Informally the problem is the following: given a set of sample input vectors and a set of sample parities, find the bits of the input vectors on which the parities were computed. In the absence of noise this problem has a known polynomial solution. In the presence of a moderate amount of noise (*i.e.,* corruption of around 1/4 of the parity bits), however, there is reason to believe that the problem is quite hard.

[1] An MDP problem is defined by choices from three distributions. For one of these distributions it is randomly self-reducible. For the other two distributions we expect MDP to be hard for other reasons (discusses in section 3).

In this paper we formally define the parity learning problem and give evidence for why it may be a hard problem. We overview the relevant work in coding theory and learning theory. We then give the translation to satisfiability used to create the parity problems in the Dimacs benchmark set. Finally, we provide experimental results for two simple satisfiability algorithms (depth-first search with unit-resolution, and iterative-sampling with unit-resolution).

2 Problem statement

Throughout this paper, the arithmetic operations of addition and multiplication are assumed to be over the field GF(2), i.e., the field of integers modulo 2. Thus, if $a, b \in \{0,1\}$ then $a+b$ is the parity of a and b. Moreover, if $\mathbf{a} \in \{0,1\}^n$ then the inner product $\mathbf{a} \cdot \mathbf{x} = \sum_{i=1}^{n} a_i x_i$ is equal to the parity of a subset of the variables $x_1, \ldots, x_n$ (specifically, the subset of variables x_i for which $a_i = 1$). It is functions of this type that we consider in this paper. Specifically, we are interested in the problem of finding the parity function that is maximally consistent with a sample of input/output pairs. This problem, which we call the minimal disagreement parity problem (MPD) is defined formally below:

Instance:

Sample Inputs: $\mathbf{x}_i \in \{0,1\}^n$ for $i = 1, \ldots, m$.

Sample outputs: $y_i \in \{0,1\}$ for $i = 1, \ldots, m$.

Error tolerance: integer k.

Find: a vector $\mathbf{a} \in \{0,1\}^n$ such that the parity function represented by **a** disagrees with the input out sample on at most k points, i.e., for which

$$|\{i : \mathbf{a} \cdot \mathbf{x}_i \neq y_i\}| \leq k.$$

Here the $\mathbf{x}_i$ represent m sample inputs over n Boolean variables, the y_i the (potentially incorrect) value of the unknown parity function on each of the m inputs, and k represents a bound on the acceptable number of errors.

The hard part of this problem is to identify the subset of the n Boolean variables over which the parity function is computed. This is represented by the **a**: for those variables i that are part of the parity function, $a_i = 1$ (and for those that are not $a_i = 0$). Of course it is not necessary to identify the correct set of variables, as long as the parity function over the set that is identified gets a sufficient number of the samples (*i.e.*, all but k of them) correct.

In this paper, we consider instances of MDP that are generated at random by the following process (where n, m and k are given):

1. First, a random *target vector* $\mathbf{s} \in \{0,1\}^n$ is generated uniformly at random.

2. Next, a set of sample inputs $\mathbf{x}_i \in \{0,1\}^n$ are generated uniformly at random for $i = 1, \ldots, m$.

3. Finally, a set of "noise bits" $r_i \in \{0,1\}$ are generated by randomly choosing exactly k of these bits to be 1, and the rest are set to 0. Each sample output y_i is then computed as $y_i = \mathbf{s} \cdot \mathbf{x}_i + r_i$.

The resulting instance is then given by the sample inputs $\mathbf{x}_i$, the sample outputs y_i and the error tolerance k.

3 Theoretical evidence for the hardness of the parity problem

In this section, we review some of the theoretical evidence suggesting the hardness of the MDP problem.

To begin with, it is known [1] that the MDP problem is NP-complete, and so is likely to be intractable in the *worst* case. However, this fact in itself is not an indication of the hardness of solving a *random* instance of the problem since there are many problems that are known to be NP-complete, and yet are trivially solvable in the average case (see, for instance, Johnson's survey article [4]). We have no proof that the parity problem is intractable for the random distributions that we consider; for instance, it is unknown if the problem is "average-case complete" in the sense of Levin [6, 3]. Nevertheless, there is considerable circumstantial evidence that solving random instances of the problem is computationally intractable.

As noted above, the random distribution of instances of the parity problem can be decomposed into three distributions: (1) the distribution of target vectors **s**; (2) the distribution of example vectors $\mathbf{x}_i$; and (3) the distribution on noise bits r_i. For the first of these, we are able to make a "random self-reducibility" argument that the average case (in which **s** is generated uniformly at random) is as hard as the worst case in which an all-powerful adversary is allowed to choose the vector **s**. Put another way, we show that any algorithm for the MDP problem may assume without loss of generality that the target vector **s** has been generated uniformly at random.

To see that this is so, consider an instance of MDP that has been generated by an arbitrary target vector **s**, sample inputs $\mathbf{x}_1, \ldots, \mathbf{x}_m$, and noise bits $r_1, \ldots, r_m$, yielding sample outputs $y_i = \mathbf{s} \cdot \mathbf{x}_i + r_i$. We show that this instance can be reduced to one in which the generating target vector is in fact chosen uniformly at random. To make the reduction, we choose a vector $\mathbf{t} \in \{0,1\}^n$ uniformly at random, and let $\mathbf{s}' = \mathbf{s} + \mathbf{t}$ (where $+$ denotes usual vector addition over GF(2)). Then $\mathbf{s}'$ is uniformly distributed. We replace each sample output y_i by

$$y_i' = y_i + \mathbf{t} \cdot \mathbf{x}_i = (\mathbf{s} + \mathbf{t}) \cdot \mathbf{x}_i + r_i = \mathbf{s}' \cdot \mathbf{x}_i + r_i.$$

Thus, each y_i' can be computed using **t** (without knowing the target vector **s**), and, moreover, we can regard y_i' as having been generated by $\mathbf{s}'$. Finally, we note that a solution $\mathbf{a}'$ to this

transformed instance can be converted back to a solution **a** to the original instance by setting $\mathbf{a} = \mathbf{a}' + \mathbf{t}$ since

$$\mathbf{a}' \cdot \mathbf{x}_i = y_i' \Leftrightarrow (\mathbf{a} + \mathbf{t}) \cdot \mathbf{x}_i = y_i + \mathbf{t} \cdot \mathbf{x}_i \Leftrightarrow \mathbf{a} \cdot \mathbf{x}_i = y_i.$$

This argument shows that if we have an algorithm that can solve instances of MDP in which the target vector **s** is generated at random, then we can solve instance in which **s** is chosen by an adversary. We conclude that there is no method of choosing "hard" target vectors that is better than simply choosing them uniformly at random.

For a different reason, we believe that the method we use of choosing sample inputs $\mathbf{x}_i$ leads to hard instances of MDP. Specifically, by choosing the $\mathbf{x}_i$ vectors uniformly at random, we create a sample in which the solution space is extremely "flat" in the sense that, except for the single target vector **s**, *all* vectors **a** have high disagreement on the sample. More precisely, if **a** is different from **s**, then the expected number of vectors $\mathbf{x}_i$ for which $\mathbf{a} \cdot \mathbf{x}_i = y_i$ is exactly $m/2$ (where, as usual, $y_i = \mathbf{s} \cdot \mathbf{x}_i + r_i$, and r_i is a random noise bit). This follows from the fact that, for a randomly chosen vector xx_i, the probability that $\mathbf{a} \cdot \mathbf{x}_i = \mathbf{s} \cdot \mathbf{x}_i$ is exactly $1/2$ if $\mathbf{a} \neq \mathbf{s}$. Thus, no gradient descent type method is likely to be an effective tool for solving such MDP problems since, as we search through the solution space, it is impossible to tell whether progress is being made until we actually locate the target vector **s**.

Finally, evidence comes from the area of computational learning theory that the randomly generated instances of MDP that we consider are intractable. Specifically, this problem is essentially equivalent to the problem of "learning" a parity function with noise. In this setting, a (random) target vector **s** is chosen, and the learner has access to noisy random examples of the form $(\mathbf{x}, \mathbf{s} \cdot \mathbf{x} + r)$, where r is a noise bit. The goal of the learner is to identify the target vector **s**.[2] In the absence of noise (or in the presence of very low levels of noise), this learning problem can be solved by linear-algebraic techniques [2]. However, it is unknown if parity functions can be efficiently learned in the presence of a constant rate of noise (for example, in which the noise bit r is 1 with probability $1/8$). Moreover, Kearns [5] has proved that it is impossible to learn noisy parity functions using a broad class of "statistical" techniques. Since the class of techniques covered by Kearns' result includes *all* of the known techniques for learning in the presence of noise, it appears quite unlikely that noisy parity functions can be learned efficiently.

Finally, we note that the MDP problem is closely related to the problem of efficiently decoding random linear codes, which is a long-standing open problem. [1].

[2] More generally, the learner's goal may be to find a function that *approximates* the behavior of the function $\mathbf{s} \cdot \mathbf{x}$. However, it can be shown that an algorithm which achieves this goal can be converted into one that achieves the seemingly harder goal of exactly identifying the target vector **s**.

4 Encoding Parity Learning as a Satisfiability Problem

4.1 The Translation

Essentially we generate a propositional theory saying "**a** is a solution to this parity learning problem". Variables representing **a** appear explicitly in the theory generated so one can read out the "answer" by looking at the values assigned to them by a model.

The translation proceeds by the following steps:

1. Randomly choose the vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$, **s**, and $r_1, \ldots, r_m$. Set $y_i = \mathbf{s} \cdot \mathbf{x}_i + r_i$. For the problems in the Dimacs challenge we used an error rate of $1/8$.[3]

2. Generate propositional formula that "calculate" the parities of $\mathbf{a} \cdot \mathbf{x}_i$. We do this with a matrix R of propositional variables. We generate propositional formula equivalent to the following:

   For $i = 1$ to m:
   $R_0^i = y_i$
   For $j = 1$ to n: $R_i^j = R_i^{j-1} \oplus a_j * x_i^j$

   These formula force $R_i^n = 0$ iff $\mathbf{a} \cdot \mathbf{x}_i = y_i$.

3. Generate propositional formula to sum the R_i^n. We do this using matrices S (sum) and c (carry) of propositional variables. The invariant here is that for $i = 0$ to $log_2(m)$: $S_i^t = i$th bit of sum of first t terms.

   For $i = 0$ to $log_2(m) : S_i^0 = 0$.
   For $j = 1$ to m:
   $c_0^j = R_j^n$
   For $i = 0$ to $log_2(m) : S_i^j = S_i^{j-1} \oplus c_i^j$.
   For $i = 1$ to $log_2(m) : c_i^j = S_{i-1}^{j-1} \wedge c_{i-1}^j$.

   These formula force S^m to be a base two representation of $|\{i : \mathbf{a} \cdot \mathbf{x}_i \neq y_i\}|$.

4. We generate propositional formula saying "S^m is at most k. We currently choose m to be a power of two and corrupt one eighth of the parity bits. This reduces this last check to just a check that the high order bits of S^m are not set.

5 Some Experimental Results

We are in the process of running experiments using iterative sampling with unit-resolution and depth-first search with unit-resolution on these problems. This should provide a basic benchmark against which to judge the more complex algorithms developed in the challenge.

[3] In the limit the hardest problems arguably will occur when the error rate is $1/4$. However, for the "small" problems we generated for the Dimacs challenge an error rate of $1/4$ gave theories with too many models.

Acknowledgements

We would like to acknowledge Haym Hirsh for his help with developing the translation of the parity problem into a satisfiability problem.

References

[1] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24, 1978.

[2] Paul Fischer and Hans Ulrich Simon. On learning ring-sum-expansions. *SIAM Journal on Computing*, 21(1):181–192, February 1992.

[3] Yuri Gurevich. Average case completeness. *Journal of Computer and System Sciences*, 42(3):346–398, 1991.

[4] David S. Johnson. The NP-completeness column: An ongoing guide. *Journal of Algorithms*, 5(2):284–299, June 1984.

[5] Michael Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 392–401, 1993.

[6] Leonid A. Levin. Average case complete problems. *SIAM Journal of Computing*, 15(1):285–286, February 1986.