Available online at www.sciencedirect.com
ScienceDirect

Theoretical Computer Science 377 (2007) 139–150

Theoretical Computer Science

www.elsevier.com/locate/tcs

Self-improved gaps almost everywhere for the agnostic approximation of monomials

Richard Nock[a,*], Frank Nielsen[b]

[a] CEREGMIA-*UFR DSE, Université des Antilles-Guyane, Campus de Schoelcher, BP 7209, 97275 Schoelcher, Martinique, France*
[b] *SONY CS Labs (FRL), 3-14-13 Higashi Gotanda, Shinagawa-Ku, Tokyo 141-0022, Japan*

Received 12 January 2006; received in revised form 5 February 2007; accepted 8 February 2007

Communicated by O. Watanabe

Abstract

Given a learning sample, we focus on the hardness of finding monomials having low error, inside the interval bounded below by the smallest error achieved by a monomial (the best rule), and bounded above by the error of the default class (the poorest rule). It is well-known that when its lower bound is zero, it is an easy task to find, in linear time, a monomial with zero error. What we prove is that when this bound is *not* zero, regardless of the location of the default class in $(0, 1/2)$, it becomes a huge complexity burden to beat significantly the default class. In fact, under some complexity-theoretical assumptions, it may already be hard to beat the trivial approximation ratios, even when relaxing the time complexity constraint to be quasi-polynomial or sub-exponential. Our results also hold with uniform weights over the examples.

Keywords: Agnostic learning; Self-improving reductions; Monomials

1. Introduction

Let $\mathcal{C}$ be a class of Boolean formulas; any of its elements, c, can be represented as a function $c : \{0, 1\}^n \rightarrow \{0, 1\}$, where n is the number of description variables. Each of the 2^n elements of $\{0, 1\}^n$ are called observations, and the couples $(o, c(o))$ are called examples, either "positive" $(+)$ when $c(o) = 1$, or "negative" $(-)$ when $c(o) = 0$. The instance of the problem which interests us is a weighted learning sample consisting of two parts:

- a subset LS of $\{(o, c(o)) : o \in \{0, 1\}^n, c(o) \in \{0, 1\}\}$, called the *learning sample*, in which the labels are given by an unknown formula c called the target concept;
- a real *weight function* $w : LS \rightarrow (0, 1]$, which assigns a weight to each example, such that all weights sum to one over LS.

[*] Corresponding author. Tel.: +596 596 72 74 28; fax: +596 596 72 74 03.
E-mail addresses: Richard.Nock@martinique.univ-ag.fr (R. Nock), Nielsen@csl.sony.co.jp (F. Nielsen).
URLs: http://www.univ-ag.fr/~rnock (R. Nock), http://www.csl.sony.co.jp/person/nielsen/ (F. Nielsen).

0304-3975/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2007.02.023

We do not know neither c, nor $\mathcal{C}$, yet our objective is to approximate as best possible c on LS using another function called a *hypothesis*; more precisely, given LS and a class of Boolean formulas $\mathcal{H}$ called the hypothesis class, our objective is to build some $h \in \mathcal{H}$ with error $err_h(LS)$ as close as possible to the best possible on $\mathcal{H}$,

$$err_*(LS) = \inf_{h \in \mathcal{H}} err_h(LS). \tag{1}$$

Here, $err_h(LS) = \sum_{(o,c(o)) \in LS : h(o) \neq c(o)} w((o, c(o)))$ is the sum of weights of the examples of LS on which h and c disagree. As long as $\mathcal{H}$ is powerful enough to express all 2^{2^n} concepts, the solutions to (1) are the concepts of $\mathcal{H}$ consistent with LS (making no error). However, seldom are the cases where such an assumption can be made, either because restrictions are put on h (e.g. for experimental purposes, see [17]), or because $\mathcal{H}$ is inherently small. The task returns then to that of minimizing the error $err_h(LS)$, or equivalently maximizing the accuracy, $acc_h(LS) = 1 - err_h(LS)$, so as to come as close as possible to the optimum ($err_*(LS)$ or $acc_*(LS)$, respectively). Solving (1), or approximating its solution, is a crucial problem from the standpoints of different learning theories, as it constitutes the core for learning various classes of Boolean formulas $\mathcal{H}$ (including the one we consider in this paper), [1,2,12,15,17,18].

A class of Boolean formulas $\mathcal{H}$ well studied in computational learning theory is the class of *monomials*, $\mathcal{M}$. A monomial is a conjunction of literals (i.e. the assignment of Boolean variables). This class is among the simplest in description, yet it encodes an exponential number of Boolean concepts (in fact, $3^n + 1$ [1]). It has received extensive coverage for both theoretical and experimental works related to machine learning and computational learning theory [16]. The essential positive result, one of the oldest in computational learning theory, states that there exists a polynomial time algorithm which, whenever $err_*(LS) = 0$, returns a monomial h with $err_h(LS) = 0$ [16]. This algorithm turns out to be linear in all parameters, and it even works when fed with the positive examples of LS only. The simplicity of the task contrasts with its hardness when $err_*(LS) \neq 0$. Indeed, if $P \neq NP$, no polynomial time algorithm can guarantee to find $h \in \mathcal{M}$ satisfying $err_h(LS) \leq err_*(LS) + \epsilon$, for some $\epsilon > 0$. This is not exactly the same formulation as in [15], but this one holds as well. Here, ϵ is a decreasing function of n, and thus the result might look not so negative for problems with large number of description variables. Unfortunately, there is more: for any $\epsilon > 0$, if $P \neq NP$, no polynomial time algorithm can guarantee to find $h \in \mathcal{M}$ satisfying

$$(K - \epsilon) acc_h(LS) \geq acc_*(LS), \tag{2}$$

where K is a constant that has been regularly improved: $K = 770/767$ in [2], $K = 59/58$ in [4] and $K = 2$ in [6]. All these results amount to saying that no polynomial time algorithm can guarantee to find $h \in \mathcal{M}$ satisfying:

$$err_h(LS) \leq (1 - \epsilon) \times err_*(LS) + \epsilon \times 1, \quad \forall \epsilon < 1 - 1/K. \tag{3}$$

(3) brings an intractable region in the interval of errors, $[err_*(LS), 1]$, as a function of its bounds' weighted arithmetic average. Since it helps to locate intractability, such a bound is very appealing from the conceptual standpoint. Since ϵ cannot be larger than half a percent, the bound is however not really meaningful from the experimental standpoint, and it leaves room for improvements from the theoretical standpoint as well. From both standpoints, the interval $[err_*(LS), 1]$ is also not the best to use. Let $\mu_*(LS) = \min\{w^+, w^-\}$, with w^+ (resp. w^-) the sum of weights of the examples from class + (resp. −). Then, the interval $[err_*(LS), \mu_*(LS)]$, where $\mu_*(LS)$ turns out to be the error of the *default class*, is much more meaningful. Indeed, its lower bound is the theoretical *best* case, while its upper bound is the experimental *worst* case, the reference in machine learning [13]. Provided $\mathcal{H}$ encodes the default class, and it is the case for $\mathcal{M}$, any learning algorithm for $\mathcal{H}$ whose output would not beat $\mu_*(LS)$ would indeed be completely useless. To see that $\mathcal{M}$ encodes the default class, notice that the empty monomial achieves error w^-, while the monomial with all $2n$ literals achieves error w^+.

In this paper, we focus on the hardness of coming close to $err_*(LS)$ in the interval $[err_*(LS), \mu_*(LS)]$, as a function of $err_*(LS)$ and $\mu_*(LS)$. Our results establish a sharp complexity plateau for this approximation. While finding $h \in \mathcal{M}$ with $err_h(LS) = 0$ costs almost nothing when $err_*(LS) = 0$ (it is linear time and only a part of the examples are sufficient, regardless of $\mu_*(LS)$), we show that when $err_*(LS) \neq 0$, significantly beating $\mu_*(LS)$ is hard. This hardness should be appreciated in the light of the corresponding consequences of our result:

[1] All monomials having at most n literals, plus the `false` concept, coded e.g. by all $2n$ variables.

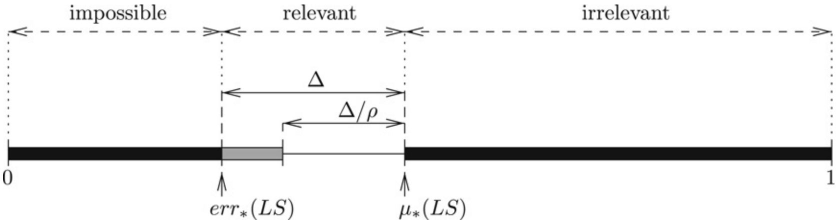

Fig. 1. An algorithm for (f, ρ, μ)-approximability outputs a monomial h whose error lies in the grey-shaded area (here, $\Delta = \mu_*(LS) - err_*(LS)$, see text for details).

- the result also holds when allowing the induction algorithm to be quasi-polynomial time, or even sub-exponential time;
- modulo some complexity assumptions, it may even be hard to beat significantly the trivial approximability ratios;
- the result still holds regardless of the location of $\mu_*(LS)$ in $(0, 1/2)$;
- the result already holds when the examples have uniform weights: thus, the weights are not the "key" to hardness. This property is particularly important since skewing the weights is the usual ingredient for obtaining or improving hardness results in learning [15,17].

It is proven via self-improving reductions, a tool that has been previously used very sparsely in computational learning theory, yet that is sometimes very helpful to bring strong inapproximability results [10,19,20]. To the best of our knowledge, our result establishes to date the sharpest complexity-theoretical plateau in settings related to agnostic approximation – finding low error classifiers when no assumption is made about the target concept to model –. It also has a consequence on Boosting using monomials as weak classifiers [16]. In this classification setting, moderately accurate classifiers are sought via an iterative reweighting scheme; all these classifiers bring altogether a combination of low error. Our results tend to be an advocacy for previous weak learning algorithms, that basically rely on an exhaustive search in a class of small cardinality, to find its best classifier [8,22].

Section 2 presents our model of agnostic approximation. Section 3 states and proves our main results, and presents related consequences. A last section concludes the paper. An Appendix contains some proofs and a background on inapproximability.

2. Agnostic approximation

Before stating our results, we first give some definitions related to approximation and learnability. They are derived from robust/agnostic learning [12], with a difference with usual learning models in that "learning" in our case is evaluated with respect to LS only, and not with respect to a potentially much larger domain from which LS is sampled. This approach evacuates statistical problems inherent to learning [16], yet it does not prevent to cast the negative results on these larger, unrestricted domains, with standard randomization/simulation arguments [14]. First, for any weighted learning sample LS, we say that LS is μ-robust as a shorthand for the fact $\mu_*(LS) = \mu$. The following definition presents the approximation of μ-robustness; it makes use of the notation $|I|$, the general notation for a problem's instance size ($|.|$ denotes the size).

Definition 1. We say that $\mathcal{H}$ is approximable in time $f(|I|)$ within ρ on μ-robust samples (in short, $\mathcal{H}$ is (f, ρ, μ)-approximable) iff there exists an algorithm $\mathcal{A}$ with the following properties:

(1) $\mathcal{A}$ runs in time $\mathcal{O}(f(|I|))$;
(2) $\mathcal{A}$ takes as input a μ-robust weighted learning sample LS, and outputs $h \in \mathcal{M}$ such that:

$$err_h(LS) \leq \frac{1}{\rho} err_*(LS) + \left(1 - \frac{1}{\rho}\right) \mu_*(LS). \quad (4)$$

In this definition, we have not put parameters in $f(.)$, since the complexity can be measured using various instance parameters. Fig. 1 gives a schematic representation of (f, ρ, μ)-approximability. The "irrelevant" region represents those monomials whose error would be larger than that of the default class, thus being irrelevant with respect to learning, while the "impossible" region represents errors unachievable by monomials. $\mathcal{A}$ is as better as ρ is close to

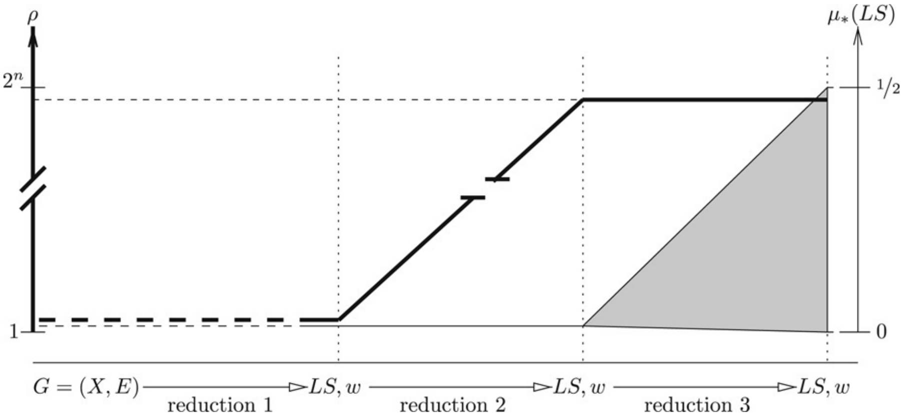

Fig. 2. Synthesis of the three reductions' effects on ρ and $\mu_*(LS)$ (see text for details).

one, and as worse as ρ increases. To finish up with the definitions, we define QP as the class of problems admitting deterministic algorithms running in quasi-polynomial time, i.e. $f(|I|) = |I|^{\text{polylog}(|I|)}$, with polylog$(|I|)$ denoting $\log^{\mathcal{O}(1)} |I|$ functions (with "log" the base-2 logarithm throughout this paper). We also consider sub-exponential time algorithms, for which $f(|I|) = 2^{|I|^\eta}$ (with $\eta < 1$, [5]). Such relaxed time constraints are authorized by very strong complexity assumptions found in various papers [21,5] (among others).

From the results in [2,15,16], it appears that the hardness of (f, ρ, μ)-approximability depends on the location of $err_*(LS)$. Indeed, whenever $err_*(LS) = 0$, simple algorithms exist for which $\rho = 1$, regardless of $\mu_*(LS)$. On the other hand, when $err_*(LS) > 0$, ρ is strictly greater than one for at least one possible value of $\mu_*(LS)$. In what follows, we make this observation more precise, and prove that when $err_*(LS)$ is non-zero, we can make ρ blow-up to virtually any intractability ratio, for any $\mu_*(LS) \in (0, 1/2)$.

3. Main result

Our proof is complexity-theoretic, and relies on the combination of three reductions. The first establishes a small inapproximability ratio ρ for some particular value of $\mu_*(LS)$; the second is a self-improving reduction (bootstrapping): it combines instances of the same problem to create a new instance that boosts ρ to very large values, still for some particular value of $\mu_*(LS)$; the third is an offset gadget that makes it possible to move $\mu_*(LS)$ everywhere in $(0, 1/2)$, while keeping the boosted inapproximability ratio ρ. Fig. 2 presents a synthesis of the combined effects of the reductions on the parameters ρ and $\mu_*(LS)$, and Fig. 3 presents the learning samples built throughout the three reductions.

3.1. First reduction: The basic step

The instance of the Max-Independent-Set problem [9] is a simple graph $G = (X, E)$, its feasible solutions are independent sets of G, i.e. any subset $S \subseteq X$ such that no edge of E has its two endpoints in S. Its objective is to maximize the size of the independent set built. Our starting point is a well known reduction from the Max-Independent-Set problem [17]. We briefly state it for the sake of completeness. Let $G = (X, E)$ be the graph instance of Max-Independent-Set, and s_* the size of its largest independent set. We create LS over $n = |X|$ variables, with $|X|$ positive examples and $|E|$ negative examples. Each variable is in one-to-one relationship with a vertex of G. The positive examples are defined as $p_i = (0_i, 1)$ for each $i \in \{0, 1, \ldots, |X| - 1\}$, where 0_i denotes the all-1 observation with a zero (negative literal) for variable i. The negative examples are defined as $n_{i,j} = (0_{i,j}, 0)$ for each edge $(i, j) \in E$, where $0_{i,j}$ denotes the all-1 observation with zeros in variables i and j. The weight of each example is $1/(|X| + |E|)$. This reduction has the following properties. Let T denote any monomial. With respect to LS, we can suppose without loss of generality that T is monotone (i.e. it does not contain negative literals). Otherwise:

- either it contains at least two negative literals, and no positive example can satisfy it. Replacing T by the all-1 monotone monomial cannot increase its error, since no negative example can satisfy it;

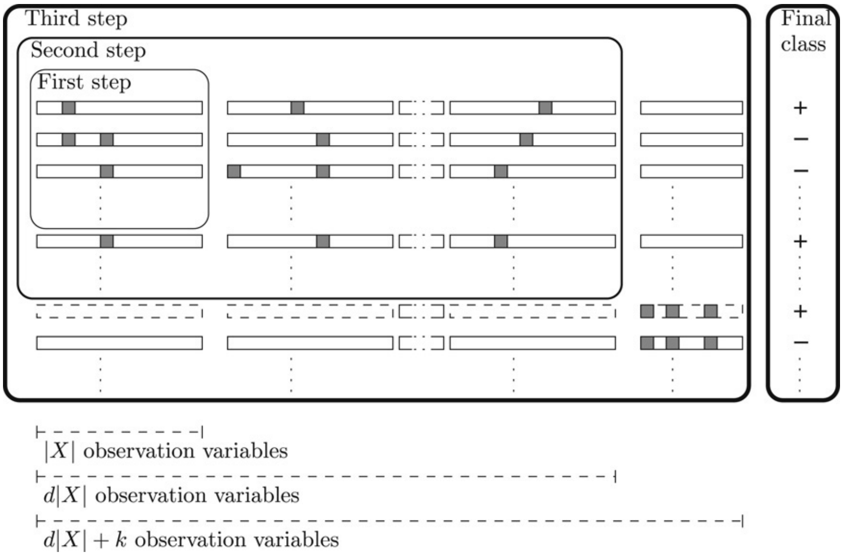
Third step
Second step
First step
Final class
+
−
−
+
+
−
$|X|$ observation variables
$d|X|$ observation variables
$d|X|+k$ observation variables

Fig. 3. An overview of examples generated through all three reductions (shaded squares denote negative literals; all others are positive. See text for details).

- either it contains exactly one negative literal, say for variable i. In this case, only p_i satisfies T among all positive examples. Replacing T with the monotone monomial in which only positive literal i is absent still makes T satisfied by p_i, and T cannot be satisfied by any negative example: its error cannot increase.

We can also suppose that T does not make errors over negative examples. Otherwise, if, say, $n_{i,j}$ satisfies T, we put the positive literal of variable i in T: now, T gives the wrong class to a positive example, but gives the right class to at least one more negative example; therefore, its error does not increase. Finally, T immediately encodes an independent set of G: take the set $S \subseteq X$ whose vertices correspond to positive literals absent from T. It cannot contain both endpoints of some edge $(i, j) \in E$, since otherwise negative example $n_{i,j}$ would satisfy T. Similarly, given any independent set of size s in G, it is easy to build a monotone monomial making no more than $(|X| - s)/(|X| + |E|) = \mu_*(LS) - s/(|X| + |E|)$ errors, by putting as positive literals all but those corresponding to vertices in the independent set. Here, we have used the fact that provided $|E| \geq |X|$,

$$\mu_*(LS) = \frac{|X|}{|X| + |E|}. \tag{5}$$

Putting this altogether, we see that the minimal error over LS and the size of the largest independent set of G are related as follows:

$$err_*(LS) = \mu_*(LS) - \frac{s_*}{|X| + |E|}. \tag{6}$$

While this is enough to translate any hard gap (thus, inapproximability ratio, see Appendix A.2) from Max-Independent-Set to the hardness of (f, ρ, μ)-approximability, this is clearly not enough to obtain large inapproximability ratios.

3.2. Second reduction: Self improvement

Now, we show how to boost the inapproximability ratio ρ. Fix $d > 1$ some integer. The reduction starts from the sample LS of Section 3.1. From this set of examples, we create new examples over $d|X|$ variables (see Fig. 3). This new set of variables is partitioned into d subsets. Each of these subsets is in bijection with the original $|X|$ variables of the first step. Any new observation can thus be described as a d-tuple, each tuple corresponding to one of these subsets, and written over the original $|X|$ variables. Positive and negative examples are created as follows:

- the observation of each new positive example is of the form $(0_{i_1}, 0_{i_2}, \ldots, 0_{i_d})$, where 0_{i_k} is the observation of positive example p_{i_k} of the first step. Thus, out of the $d|X|$ variables, the observation of each new positive example contains d negative literals. Making all possible combinations yield $|X|^d$ new positive examples. Let $p_{i_1,i_2,\ldots,i_d}$ be the notation for the new positive examples;

- the observation of each new negative example is obtained via the following transformation: we take some observation $(0_{i_1}, 0_{i_2}, \ldots, 0_{i_d})$ of positive example $p_{i_1,i_2,\ldots,i_d}$, and replace 0_{i_k} (for some $1 \leq k \leq d$) by some $0_{i,j}$, where $0_{i,j}$ is the observation of negative example $n_{i,j}$ of the first step. Thus, the observation of this new negative example, noted $(0_{i_1}, 0_{i_2}, \ldots, 0_{i_{k-1}}, 0_{i,j}, 0_{i_{k+1}}, \ldots, 0_{i_d})$, contains exactly $d + 1$ negative literals. We let $n_{i_1,i_2,\ldots,i_{k-1},(i,j),i_{k+1},\ldots,i_d}$ denote this negative example. Making all possible transformations yield exactly $d|E||X|^{d-1}$ distinct observations, as the observation of any $p_{i_1,i_2,\ldots,i_d}$ exactly brings $d|E|$ distinct observations, but each distinct observation obtained via the transformation can be obtained from $|X|$ distinct new positive example ($p_{i_1,i_2,\ldots i_{k-1},i_k,i_{k+1},i_d}$ yield $n_{i_1,i_2,\ldots,i_{k-1},(i,j),i_{k+1},\ldots,i_d}$, for any $i_k = 0, 1, \ldots, |X| - 1$). Overall, we create all these $d|E||X|^{d-1}$ new negative examples.

Weights are uniform, i.e. equal to

$$w = \frac{1}{|X|^d + d|E||X|^{d-1}}. \tag{7}$$

For any monomial T, let $(T_1, T_2, \ldots, T_d)$ be a d-tuple to represent T over the partition of variables adopted for the representation of the examples. Each T_i is thus described over the initial set of $|X|$ variables, and any observation $(o_1, o_2, \ldots, o_d)$ satisfies T iff o_j satisfies T_j, for any $j = 1, 2, \ldots, d$. W.l.o.g., we can suppose that any monomial T built over this new set of examples is monotone. Otherwise, we can transform it in $\mathcal{O}(d|X|)$ time so as to make it monotone, as follows. Suppose that some T_i is not monotone:

- if T_i contains more than one negative literal, T cannot be satisfied by any positive example. We can thus replace T_i by the all-1 monomial without increasing the error;
- if T_i contains exactly one negative literal, say for variable j, then we remove this negative literal and complete T_i with all positive literals for all *other* variables. This procedure keeps the same number of positive examples that satisfy T, while it cannot increase the number of negative examples that satisfy T. Thus, the error of T does not increase.

Lemma 2. *Without loss of generality, T does not misclassify any negative example.*

(Proof in appendix, Appendix A.1). Now, a glimpse at the structure of T reveals that each of its parts over the $|X|$ initial variables encodes an independent set on the original graph, which corresponds to the literals absent in the part; furthermore, while T gives the right classification to all negative examples, its accuracy over the positive examples is simply the (weighted) product of the number of "holes" (unspecified variables) over each of the d copies of the original $|X|$ variables. Reciprocally, given a feasible solution to MAX-INDEPENDENT-SET, it is easy to build a monotone monomial T by concatenating in each of the d copies of the variables the monomial described over the $|X|$ initial variables in Section 3.1 : T errs only on positive examples, corresponding to those for which at least one of their d negative literal is not in the unspecified variables of T. We obtain:

$$err_*(LS) = \mu_*(LS) - \frac{s_*^d}{|X|^d + d|E||X|^{d-1}}, \tag{8}$$

where we recall that s_* is the MAX-INDEPENDENT-SET maximal solution's size, and, provided $|E| \geq |X|/d$:

$$\mu_*(LS) = \frac{|X|}{|X| + d|E|} \tag{9}$$

is the total weight for the positive class. Now, this is enough to boost any hard gap (thus, inapproximability ratio) for MAX-INDEPENDENT-SET to a power of d for (f, ρ, μ)-approximability. However, since the reduction's time is also $\mathcal{O}(|X|^d)$, one has to take care of choosing d not too large to make the final time complexity "fit" into the assumption used. We shall see it later. In the final step, we see how to release $\mu_*(LS)$ from (9), and make it shift anywhere into $(0, 1/2)$.

3.3. Third reduction: The offset gadget

Here, we show that the hard gap of Section 3.2 can be kept even when moving the default class error along the interval (0, 1/2). The principle is to add another gadget construction in the variables and examples after the second reduction. Let $\mu_0 = \mu_*(LS)$ as defined in (9).

Suppose that we want to increase $\mu_*(LS)$ to some $\mu_1 = \frac{a}{2a+b} \in (\mu_0, 1/2) \cap \mathbb{Q}$, where $\mathbb{Q}$ denotes the set of rationals, and $a, b \in \mathbb{N}^*$ are any two constants. We suppose without loss of generality that the graph instance of Max-Independent-Set has $|X|$ and $|E|$ multiples of b. This is not a restriction for hardness results, since making b copies of the same graph, without adding edges between different copies, satisfies the two conditions while it also multiplies the size of the maximal independent sets by b. Hence, inapproximability ratios are preserved (see Appendix A.2). Let us fix:

$$m = (ad|E||X|^{d-1} - (a+b)|X|^d)/b, \tag{10}$$

$$k = \lceil \log(m+1) \rceil = \mathcal{O}(d \log |X|). \tag{11}$$

Notice that the assumption $\mu_1 > \mu_0$ implies $m > 0$. What we now do is a simple trick that shifts Bayes error for LS (the minimal error on LS over *all* possible classifiers, not restricted to monomials). We create k new variables in addition to the last $d|X|$. The examples obtained in reduction two (hereafter referred to as "old" examples) all have the same value for the new variables, namely we put only positive literals for these variables (see Fig. 3). We also create $m \le 2^k - 1$ *new* positive examples: each of them contains, for the first $d|X|$ variables, all positive literals, and we plug for the last k variables one of the m choices for the literals of the last variables *except* the choice of all positive literals. We also create m new negative examples, each of which has exactly the *same* observation of some new positive example. Again, weights are uniform. We let $T = (T_1, T_2, \ldots, T_d, T_\star)$ denote some optimal monomial for this last reduction, using the convention of the two last reductions, with $T_\star$ built over the last k variables.

Lemma 3. *Without loss of generality, T is not satisfied by any of the m new positive examples.*

The proof of this lemma is immediate once we remark that if any new positive example satisfies T, then so does one new negative example. Thus, if we replace $T_\star$ by all positive literals of the last k variables, then we do not increase the error of T. Since the negative class is still the majority class, we have:

$$\mu_1 = \frac{\mu_0}{w} \times \frac{1}{\frac{1}{w} + 2m} + \frac{m}{\frac{1}{w} + 2m} = \frac{a}{2a+b}. \tag{12}$$

Following the second reduction, since all new positive examples receive the wrong class (Lemma 3), we obtain that $err_*(LS)$ satisfies:

$$err_*(LS) = \mu_1 - \frac{s_*^d}{|X|^d + d|E||X|^{d-1} + 2m} = \mu_1 - \frac{\kappa s_*^d}{|X|^d + d|E||X|^{d-1}}, \tag{13}$$

with:

$$\kappa = \frac{1}{1 + 2wm} = \theta(1), \tag{14}$$

while the whole reduction time is no more than $\mathcal{O}(d|X|^{d+1})$.

Should we have needed to decrease μ_0 to some $\mu_1 = \frac{a}{2a+b} \in (0, \mu_0) \cap \mathbb{Q}$, we would only have added the new negative example for the third reduction (this time without changing Bayes error), with:

$$m = ((a+b)|X|^d - ad|E||X|^{d-1})/a, \tag{15}$$

and k fixed as in (11), this time with the assumption that $|X|$ is a multiple of a.

3.4. Consequences

The most immediate consequence relies on the following theorem, whose proof is postponed to Appendix A.2, which also contains the necessary background on inapproximability.

Theorem 4 (*Hardness of* (f, ρ, μ)*-Approximability*). *For any* $\mu \in (0, 1/2) \cap \mathbb{Q}$, $\mathcal{M}$ *is not* (f, ρ, μ)*-approximable, for:*

- $\rho = n^K$, *for any constant* $K > 0$, *unless* $NP = P$ *(and* f *is polynomial);*
- $\rho = n^{\log^K n}$, *for any constant* $K > 0$, *unless* $NP \subset QP$ *(and* f *is quasi-polynomial);*
- $\rho = 2^{n^K}$, *for some constant* $K < 1$, *unless* $NP \subset DTIME(2^{|I|^\eta})$ *for some* $\eta < 1$ *(and* f *is sub-exponential).*

Furthermore, the result holds even with uniform weights over the examples.

Notice that the constraint that $\mu_*(LS)$ is rational is not a restriction, since $\mathbb{Q}$ is dense into $(0, 1/2)$. Up to what is hidden in the K, its inapproximability ratio 2^{n^K} may not be far from the trivial ratio, 2^n. Indeed, whenever $\mu_*(LS) = w^+$, if we replace `false` by the observation of the positive example having the largest weight, then the new error is $\leq w^+ - (1/2^n) < w^+(1 - (1/2^n)) < (1/2^n)err_*(LS) + (1 - (1/2^n))\mu_*(LS)$.

As a comparison, negative results obtained in the framework of agreements maximization in (2) can be translated to negative results for (f, ρ, μ)-approximability, but at the expense of a weakening of the inapproximability ratio. Consider for example the best of these results, in [6]. Here, a hard gap is obtained for monomials with a reduction which brings a learning sample LS with uniform weights, with $\mu_*(LS) = 1/2$, and for which the error of the optimal monomial either satisfies $err_*(LS) \leq \epsilon/2$, or $err_*(LS) > 1/2 - \epsilon/2$, for any constant $\epsilon > 0$. This implies (f, ρ, μ)-inapproximability for any ρ such that $err_*(LS)/\rho + (1/2)(1 - 1/\rho) \leq 1/2 - \epsilon/2$, i.e. such that $\rho \leq 1/\epsilon - 1$. Thus, the best possible inapproximability ratios for (f, ρ, μ)-approximability that follow from [6] are constant. Moreover, using the inapproximability ratios obtained by [6] with the power of randomized quasi-polynomial time reductions would only yield sublinear inapproximability ratios for (f, ρ, μ)-inapproximability, i.e. with $\rho < n$.

A consequence of Theorem 4 is the following:

Corollary 5. *unless* $NP = P$, *for any* $\delta \in (0, 1/2)$, *there does not exist some polynomial time algorithm that, given learning sample* LS *and weight function* w, *returns when there exists one some* $h \in \mathcal{M}$ *with* $err_h(LS) \leq 1/2 - \delta$. *The result remains true if we pick* $\delta = 1/p(n)$, *for any polynomial* $p(n)$.

(Proof in appendix, Appendix A.3). Notice that we can replace the polynomial time complexity by quasi-polynomial time if we replace the complexity assumption by $NP \subset QP$, and by sub-exponential time if we replace it by $NP \subset DTIME(2^{|I|^\eta})$. An interesting application of Corollary 5 lies in the framework of Boosting. Boosting relates to a breakthrough in learning, that has formalized the idea of combining many moderately accurate weak hypotheses to get a strong combined classifier [7,16]. Informally, suppose we have a polynomial time algorithm *weak learning* algorithm for $\mathcal{H}$, i.e. an algorithm that, given any LS and any weight function w over LS, outputs a hypothesis $h \in \mathcal{H}$ γ-far from random, i.e. with $|(1/2) - err_h(LS)| \geq \gamma$ for some $\gamma \in (0, 1/2)$. Then, there exists a polynomial time *strong learning* algorithm that, given the sole access to the weak learning algorithm, takes as input some LS and w, and returns, after T weak learning calls, a combination h_{comb} of the T outputs having error $err_{h_{\text{comb}}}(LS) \leq \exp(-T\gamma^2/2)$. By means of words, weak learning is enough to obtain, by subtle combinations, an overall error rapidly vanishing with T. So far, some authors have proned the use of simple weak hypotheses, such as decision stumps [8]. A decision stump is a decision tree with a single internal node, i.e. it is almost a monomial with a single literal. The interest in using this simple class is that the brute force approach that explores all hypotheses to find the optimal one on LS with w is linear, though its drawback is that we cannot hope to have a large γ. Replacing stumps by monomials would be more convenient from the γ standpoint, since the average error of stumps is $1/2$ regardless of $\mu_*(LS)$, while the average error of monomials satisfies:

$$\begin{aligned}\left|\frac{1}{2} - \frac{1}{|\mathcal{M}|}\sum_{h \in \mathcal{M}} err_h(LS)\right| &= \left(\frac{1}{2} + \frac{2^n}{3^n + 1}\right)(1 - 2\mu_*(LS)) \\ &= (1 + o(1)) \times ((1/2) - \mu_*(LS)),\end{aligned} \tag{16}$$

which is $\neq 1/2$ whenever $\mu_*(LS) < 1/2$, and can even be very far from $1/2$ when $\mu_*(LS)$ gets small. The proof of this statement is immediate once we remark that every positive example gets misclassified by exactly $3^n + 1 - 2^n$ monomials, while every negative example gets misclassified by exactly 2^n monomials. Thus, there almost always exist monomials γ-far from random, for large γ. The problem is that the brute-force exploration of $\mathcal{M}$ is intractable, and furthermore Corollary 5 brings that unless widely believed complexity assumptions are false, no weak learning algorithm exists for $\mathcal{M}$, for any $\gamma \in (0, 1/2)$, and we can even pick γ inversely polynomial, thus obtaining negative results in the original weak learning framework [16]. From the computational standpoint, this is an advocacy for using in weak learning the largest classes that remain tractable for the brute-force search of their optimal element, such as classes of classifiers whose elements have size bounded above by a constant [8,22].

We can also refine our hardness results to show that hardness already holds whenever there are only few examples. Berman and Karpinsky [3] build a MAX-INDEPENDENT-SET instance on a regular graph of degree three, with $284u$ nodes, for which the maximum independent set size is either above $(140-\epsilon)u$ or below $(139+\epsilon)u$ (with $\epsilon \in (0, 1/2)$). This is enough to prove that MAX-INDEPENDENT-SET is not approximable to within $140/139 - \epsilon$, for any $\epsilon > 0$. In our case, we see that the reduction of Section 3.1 brings a LS having no more than $|X| + |E| = 5|X|/2 = 5n/2$ examples. Thus, (f, ρ, μ)-approximability (with $\mu_*(LS) = 2/5 = 40\%$ in this particular case) is already hard when there are as few examples as $|LS| = \mathcal{O}(n)$, for the same inapproximability ratio as [3]. Notice that our inapproximability ratio is constant in this case, which also follows from the reduction of Feldman [6] (see above), but even when $|LS| = \mathcal{O}(n)$ in Feldman [6], the hidden constant appears to be much larger than ours.

4. Conclusion

The reduction we have used in Section 3.2 is a self-improving reduction. Basically, it can be viewed as a parameterized reduction from a problem onto itself, for which the inapproximability ratio behaves as an *exponential* function of the parameter (d in our proofs). When this parameter increases, the time complexity of the reduction increases, but the inapproximability ratio is blown up. The point is to obtain the largest possible inapproximability ratio, while keeping the reduction's time complexity inside the admissible bounds, given that these bounds are led by the complexity assumption used. Such self-improving reductions have been previously used very sparsely in computational learning theory [10,19,20]. With this tool, we have been able to show a quite striking phenomenon. Beating the majority class with monomials is a huge complexity burden whenever $err_*(LS) \neq 0$, regardless of the location of the default class error; it may even be the case for beating the trivial *exponential* approximability ratios with sub-exponential time algorithms. In deep contrast, when $err_*(LS) = 0$, achieving *optimal* error, with *unit* approximability ratio, costs only a linear-time machinery which does not even make use of all examples! To the best of our knowledge, this is the sharpest complexity plateau that has appeared so far in complexity-theoretical approaches to computational learning theory.

Acknowledgments

We would like to thank the reviewers for insightful comments that helped to improve significantly the paper.

Appendix A.

A.1. *Proof of Lemma 2*

We show that we can suppose that T does not misclassify any negative example; otherwise, we transform it in linear time without increasing the error, as follows. Suppose $T = (T_1, T_2, \ldots, T_d)$ is satisfied by $n_\star = n_{i_1,i_2,\ldots,i_{k-1},(i,j),i_{k+1},\ldots,i_d} \in LS$. Let $LS_i^+, LS_i^- \subset LS$ be defined as:

$$LS_i^- = \left\{ n_{i'_1,i'_2,\ldots,i'_{k-1},(i,j'),i'_{k+1},\ldots,i'_d} \in LS : n_{i'_1,i'_2,\ldots,i'_{k-1},(i,j'),i'_{k+1},\ldots,i'_d} \text{ satisfies } T \right\},$$

$$LS_i^+ = \left\{ p_{i'_1,i'_2,\ldots,i'_{k-1},i,i'_{k+1},\ldots,i'_d} \in LS : \exists n_{i'_1,i'_2,\ldots,i'_{k-1},(i,j'),i'_{k+1},\ldots,i'_d} \in LS_{i,j}^- \right\}.$$

It follows that $n_\star \in LS_i^-$, all examples of LS_i^+ also satisfy T, and *any* positive example of the form $p_{i'_1,i'_2,\ldots,i'_{k-1},i,i'_{k+1},\ldots,i'_d}$ that satisfies T is in LS_i^+ (otherwise, since $n_{i'_1,i'_2,\ldots,i'_{k-1},(i,j),i'_{k+1},\ldots,i'_d}$ satisfies T, we would have missed this example in LS_i^-). Now, put in T_k the positive literal of variable i. After the transformation, the only positive examples that do not satisfy T anymore are those of LS_i^+, while the negative examples of LS_i^- do not satisfy T anymore. Since $|LS_i^-| \geq |LS_i^+|$ (the map which associates each negative example $n_{i'_1,i'_2,\ldots,i'_{k-1},(i,j'),i'_{k+1},\ldots,i'_d}$ of LS_i^- to the corresponding positive example $p_{i'_1,i'_2,\ldots,i'_{k-1},i,i'_{k+1},\ldots,i'_d}$, in LS_i^+, is surjective), the error of T does not increase after transformation.

A.2. Background on (in)approximability and proof of Theorem 4

The complexity of approximating MAX-INDEPENDENT-SET follows immediately from that of approximating its dual, MAX-CLIQUE and the PCP characterization of NP [11,23]. Without entering into details, decision problems in NP have a proof system in which the certificate π can be checked by a randomized verifier with limited resources, and whose probability of accepting a proof has a "gap": suppose without loss of generality that π is a proof for a Boolean formula ϕ. Then if ϕ is satisfiable, then the maximum acceptance probability is 1. Otherwise, it is no more than some $s < 1$ (thus, even a good approximation of the maximum could help to decide NP-Complete problems). This hard gap between probabilities can be translated to MAX-INDEPENDENT-SET via a reduction on a graph G of $|X| = 2^{(1+\delta)R}$ vertices ($R > 0$ is a parameter and $0 < \delta < 1$ a free parameter of the reduction), such that whenever ϕ is satisfiable, there exists a clique of size 2^R, while otherwise the cliques of G have size $<2^{\delta R}$ [23]. In this proof, we have $|X| = o(|E|)$, so that all constraints of Sections 3.1–3.3 for $\mu_*(LS)$ are satisfied. The ratio $\rho = 2^R/2^{\delta R} = \left(2^{(1+\delta)R}\right)^{1-\left(\frac{2\delta}{1+\delta}\right)} = |X|^{1-\epsilon}$ is thus NP-hard to achieve, with $\epsilon = 2\delta/(1+\delta) > 0$. Zuckerman [23] has derandomized the results of Håstad [11], which allows to remove randomized complexity classes in the complexity assumptions.

Now, let us shift to (f, ρ, μ)-approximability, and consider (13). Whenever ϕ is satisfiable, this means that the optimal monomial satisfies:

$$err_*(LS) = \mu_*(LS) - \Delta, \tag{17}$$

with:

$$\Delta = \frac{\kappa|X|^{1-(\epsilon/2)}}{|X| + d|E|} \tag{18}$$

(and κ in (14)), while when ϕ is not, the optimal monomial satisfies:

$$err_*(LS) > \mu_*(LS) - \frac{\Delta}{|X|^{d(1-\epsilon)}}. \tag{19}$$

Should $\mathcal{A}$ be some (f, ρ, μ)-approximability algorithm for $\mathcal{M}$ (for some ρ), we would obtain that when ϕ is satisfiable (after having carried out all three reductions), the monomial h output by $\mathcal{A}$ would satisfy (we use (17)):

$$\begin{aligned} err_h(LS) &\le \frac{1}{\rho} err_*(LS) + \left(1 - \frac{1}{\rho}\right)\mu_*(LS) \\ &\le \frac{1}{\rho}(\mu_*(LS) - \Delta) + \left(1 - \frac{1}{\rho}\right)\mu_*(LS) = \mu_*(LS) - \frac{\Delta}{\rho}. \end{aligned}$$

Considering (19), choosing any $\rho \le |X|^{d(1-\epsilon)}$ would allow to solve NP-Complete problems with $\mathcal{A}$, in $f(.)$-time, a contradiction if $NP \not\subseteq DTIME(f(|I|))$. Hence, if we wish to prove some inapproximability ratio $\rho(n)$ for (f, ρ, μ)-approximability, we only have to tune d and k to have (for some $\epsilon > 0$):

$$|X|^{d(1-\epsilon)} > \rho(n). \tag{20}$$

Because of the third reduction, we know that if we want $\mu \in (0, 1/2) \cap \mathbb{Q}$ constant (see (12)), then we only need $k = \mathcal{O}(d \log |X|)$ in the third reduction (see (11)); thus:

- if we want $\rho = n^K$, then we only need $|X|^{d(1-\epsilon)} > (d(|X| + K' \log |X|))^K$, for some constant K'. Picking $d > \lceil (K + 1)/(1 - \epsilon) \rceil$ – a constant – is enough for this inapproximability ratio to hold. Note also that the reduction time is polynomial;
- if we want $\rho = n^{\log^K n}$, it is enough to choose $d \ge \lceil (\log^{K+1} |X|)/(1 - \epsilon) \rceil$, thus yielding a quasi-polynomial time reduction;
- finally, to get a sub-exponential ρ, notice that the overall time complexity for the reductions of Sections 3.1–3.3 is no more than $\mathcal{O}(|X|^{d+2})$, provided $d = \mathcal{O}(|X|)$. While keeping this complexity in $\mathcal{O}(2^{|I|^\eta})$, this authorizes to pick d as large as $d = |X|^{\eta - f(|X|)}$, for some $f(|X|) = o(1)$ function (we can pick e.g. $f(|X|) = (2 \log \log |X|)/ \log |X|)$. Taking logarithms in (20), the largest inapproximability ratios are then obtained by

choosing K constant as large as possible satisfying:

$$(1-\epsilon)|X|^{\eta-f(|X|)}\log|X| = \Omega\left(|X|^{K(\eta-f(|X|))}(|X|+\log|X|)^K\right). \tag{21}$$

Picking $K = \eta/(2+2\eta)$, a constant, is enough for this to hold.

A.3. Proof of Corollary 5

We first start with δ constant. By the proof of Theorem 4, no such algorithm could exist that would find, whenever possible, a monomial h whose error would lie in the "interval gap":

$$err_h(LS) \in \left[\mu_*(LS)-\Delta, \mu_*(LS)-\frac{\Delta}{\rho}\right) = \mathbb{I}, \tag{22}$$

since otherwise, it could be used to decide NP-Complete problems. What we have to do to prove Corollary 5 is thus to carry out reduction three to pick $\mu_*(LS)$ so that $1/2-\delta \in \mathbb{I}$, which is simple since $\mu_*(LS)$, Δ and ρ are functions of $\epsilon, d, |X|, |E|$ only, and $\mathbb{Q}$ is dense in $\mathbb{R}$.

Similarly, should we want this to hold for $\delta = 1/p(n)$ for some polynomial $p(n)$, we would only have to prove that reduction three still works in polynomial time for shifting μ_1 to any $1/2-1/q(n)$ for any polynomial $q(n)$, i.e. prove that it works for $a(n) = q(n)-1$ and $b = 2$. To prove this, we keep the same expression for m with the additional dependences on n, as $m = (a(d|X|+k)d|E||X|^{d-1} - (a(d|X|+k)+b)|X|^d)/b$, for a slightly larger k:

$$k = \lceil (d+\text{degree}(q)+2)\log|X|\rceil, \tag{23}$$

with degree(q) the degree of polynomial q. A new set of k variables can represent all the m new examples if we choose m assignments of the k new variables, among the $2^k > m$ possible. The whole reduction time is no more than $\mathcal{O}(d|X|^{d+1+\text{degree}(q)})$.

References

[1] P. Bartlett, S. Ben-David, Hardness results for neural network approximation problems. in: Proc. of the 4th European Conf. on Computational Learning Theory, 1999, pp. 50–62.

[2] S. Ben-David, N. Eiron, P. Long, On the difficulty of approximately maximizing agreement, in: Proc. of the 13th International Conference on Computational Learning Theory, 2000 pp. 266–274.

[3] P. Berman, M. Karpinsky, On some tighter inapproximability results, in: Proc. of the 26th International Colloquium on Automata, Languages and Programming, 1999, pp. 200–209.

[4] N. Bshouty, L. Burroughs, Maximizing agreements and coagnostic learning, Theoretical Computer Science 350 (2006) 24–39.

[5] U. Feige, A threshold of ln n for approximating set cover, in: Proc. of the 28th ACM STOC, 1996, pp. 314–318.

[6] V. Feldman, Optimal hardness results for maximizing agreements with monomials, in: Proc. of the 21th IEEE International Conference on Computational Complexity, 2006, pp. 226–236.

[7] Y. Freund, R.E. Schapire, A Decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1997) 119–139.

[8] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: A statistical view of boosting, Annals of Statistics 28 (2000) 337–374.

[9] M. Garey, D. Johnson, Computers and Intractability, a guide to the theory of NP-Completeness. Bell Telephone Laboratories, 1979.

[10] T. Hancock, T. Jiang, M. Li, J. Tromp, Lower bounds on learning decision lists and trees, Information and Computation 126 (1996) 114–122.

[11] J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, Acta Mathematica 182 (1999) 105–142.

[12] K.-U. Höffgen, H. Simon, K.-S.V. Horn, Robust trainability of single neurons, Journal of Computer and System Sciences 50 (1995) 114–125.

[13] R. Holte, Very simple classification rules perform well on most commonly used datasets, Machine Learning 11 (1993) 63–91.

[14] M. Kearns, M. Li, L. Pitt, L. Valiant, On the learnability of boolean formulae, in: Proc. of the 19th ACM Symposium on the Theory of Computing, 1987, pp. 285–295.

[15] M. Kearns, R.E. Schapire, L.M. Sellie, Toward efficient agnostic learning, Machine Learning 17 (1994) 115–141.

[16] M.J. Kearns, U.V. Vazirani, An Introduction to Computational Learning Theory, M.I.T. Press, 1994.

[17] R. Nock, Complexity in the case against accuracy estimation, Theoretical Computer Science 301 (2003) 143–165.

[18] R. Nock, P. Jappy, Function-free horn clauses are hard to approximate, in: Proc. of the 8th International Conference on Inductive Logic Programming, in: Lecture Notes in Artificial Intelligence, vol. 1446, Springer-Verlag, 1998, pp. 195–204.

[19] R. Nock, P. Jappy, J. Sallantin, Generalized graph colorability and compressibility of boolean formulae, in: Proc. of the 9th International Symp. on Algorithms and Computation, in: Lecture Notes in Artificial Intelligence, vol. 1533, Springer-Verlag, 1998, pp. 237–246.

[20] R. Nock, M. Sebban, Sharper bounds for the hardness of prototype and feature selection, in: Proc. of the 11th International Conference on Algorithmic Learning Theory, in: Lecture Notes in Artificial Intelligence, vol. 1968, Springer-Verlag, 2000, pp. 224–237.

[21] K. Pillaipakkamnatt, V. Raghavan, On the limits of proper learnability of subclasses of DNF formulae, in: Proc. of the 7th International Conference on Computational Learning Theory, 1994, pp. 118–129.

[22] R.E. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Machine Learning 37 (1999) 297–336.

[23] D. Zuckerman, Linear degree extractors and the inapproximability of max clique and chromatic number, in: Proc. of the 38th ACM Symposium on the Theory of Computing, 2006, pp. 681–690.