Learning with Maximum-Entropy Distributions*

Yishay Mansour **Mariano Schain**
Computer Science Dept.
Tel-Aviv University
{mansour,mariano}@math.tau.ac.il

Abstract

We are interested in distributions which are derived as a maximum entropy distribution given a set of constraints. More specifically, we are interested in the case where the constraints are the expectation of individual and pairs of attributes. For such a given maximum entropy distribution we develop an efficient learning algorithm for read-once DNF. We also show how to extend our results to monotone read-k DNF, following the techniques of [HM91]

1 Introduction

The PAC learning model [Val84] is the most basic model in computational learning theory. Its introduction brought forward a simple set of assumptions and raised many challenging problems. Initially, the main goal was a computational one, to develop new algorithms within this framework and show the learnability of different concept classes. The PAC model has been very successful in the study of the tradeoff between sample size versus accuracy and confidence, but less successful in the algorithmic study. Only few algorithmic techniques where developed, and many simple concept classes have been proven to be computationally intractable. To mention two intractability results in the PAC model, efficient learning 3-term DNF [PV88] (for learning using a hypothesis which is a 3 term DNF) and efficient learning polynomial size formula [KV88] (for learning using any hypothesis).

In addition, due to the fact that the distribution may be arbitrary, it was rather simple to show equivalence in the difficulty of related concept classes, such as general DNF and monotone read-once DNF [HKLW90]. A much more complicated and non-intuitive result is showing that if DNF is learnable with membership queries, in the PAC model, then DNF can be learned also without membership queries [AK91].

Researchers were aware that the need to learn a concept class with respect to an *arbitrary* distribution is one of the main sources of intractability in the PAC model. Research was done on the learnability with respect to a fixed distribution, and a model, similar to the PAC model, was suggested, where the learning algorithm may depend on the underline distribution [BI88, Nat87]. They also show examples where this allows to learn classes which where not learnable before.

When looking for learning with respect to a fixed distribution, rather than an arbitrary one, it is most important to chose "natural" distributions. The most common distributions are the uniform distribution and product distributions. For example with respect to such distributions one can learn read-once DNF [KLV94], although with respect to an arbitrary distribution it is equivalent to learning general DNF, which is a major open problem in computational learning theory.

This was the starting point of our research, we were interested in widening the set of "natural" distributions that are considered. One huge difference between an arbitrary distribution and a uniform/product distribution is the independence assumption. In a product distribution, by definition, the values of the various attributes are independent. This is a very severe handy-cap when considering "natural" distributions, and clearly helps the learning algorithm. We are interested in considering a wide class of distributions where the independence assumption does not hold. First let us deviate and discuss maximum entropy distributions.

The entropy is a functional over probability distributions, intuitively it measures the uncertainty of an observer about the outcome of an experiment (see, e.g. [CT91]). Maximizing the entropy may be viewed in many cases as trying to adapt the "most permissive" distribution that obeys certain constraints. We view our

*This research was supported in part by a grant from the Israel Science Foundation.

setting as being given a set of constraints on the distribution, each constraint is an expectation of a predicate of the attributes. Later, assuming that there are distributions that satisfy the set of constraints, we focus, on the distribution that maximizes the entropy.

In order to motivate this, let us consider a few special cases. First, if there are no constraints, then all the distributions are possible, and the maximum entropy distribution is the uniform distribution. Second, if we are given the expectations of individual attributes as constraints, and no other constraint, the maximum entropy is the product distribution, where each attribute has the desired expectation. The case that we chose to concentrate on in this work is the one where we have expectations of individual variables and pairs of variables (i.e., probability that two attributes are simultaneously one).

There is a vast literature about the subject of maximum entropy distributions, a subject of special interest in physics [SWG85]. One of the basic results is the form of the maximum entropy distribution, given a set of constraints. Unfortunately, from a computational point of view, it is NP-complete to decide if there is any distribution that satisfies the set of pair-wise constraints [KM93, KN95].

As stated before, learning the read-once DNF concept class over arbitrary distribution is equivalent to DNF [HKLW90]. Constraining the underlying probability distribution to uniform or product distribution enables to learn read-once DNF [KLV94]. However, the algorithms for learning the read-once DNF over uniform or product distributions depends heavily on the fact that the attributes are independent. This is why it is interesting to develop an efficient learning algorithm for read once DNF under maximum entropy distributions. This enables us to extend our learnability horizon and include many distribution classes which do not have the independence assumption.

We present an efficient learning algorithm for read-once DNF over the Maximum Entropy distribution for pairwise expectations. Our algorithm receives the parameters of the maximum entropy distribution, and based on them and random examples from the target concept, it finds a read-once DNF that approximates well the target read-once DNF.

We extend the rusults to monotone read-k DNF. A previous result by [HM91] describes an algorithm that efficiently approximates monotone read-k DNF over product distributions. We follow thier techniques and prove that a simmilar algorithm works for Maximum Entropy distributions as well.

Our algorithm requires only statistical queries, for this reason we use the Statistical Query model [Kea93] to describe the algorithm. We use a variant of the statistical query model where we have the ability to query on the expectation of a real valued function with a bounded range, instead of only boolean predicates. This variant preserves the fact that learning in it implies PAC learnability even in the presence of random classification noise. (We need the range of the function to be bounded by a constant in order to maintain our ability to simulate it in the PAC model with a constant additive error.) As a result, our algorithm is tolerant to random classification noise.

2 Model and notation

2.1 PAC Learning model

A concept c is a Boolean function $c : \mathcal{X} \to \{0,1\}$, where $\mathcal{X}$ is the learning domain. An example for c is a pair $(\mathbf{x}, c(\mathbf{x}))$, where $\mathbf{x} \in \mathcal{X}$.

Let D be a fixed probability distribution over $\mathcal{X}$ and let $\mathcal{C}$ be a set of concepts. A learning algorithm for concept class $\mathcal{C}$ with respect to distribution D is an algorithm that has access to examples of an unknown target concept $c \in \mathcal{C}$ drawn independently according to D. The output of the learning algorithm is a hypothesis $h \in \mathcal{C}$. The *error* of h with respect to the distribution D and the target concept c is

$$Error_D(h,c) \stackrel{\text{def}}{=} Pr_D(h \neq c)$$

Similar to [BI88], we say that an algorithm *PAC*-learns the concept class $\mathcal{C}$ over distribution D if for any $c \in \mathcal{C}$ given $\epsilon, \delta > 0$ the algorithm outputs a hypothesis $h \in \mathcal{C}$, such that

$$Pr(Error_D(h,c) \geq \epsilon) \leq \delta.$$

We say that such a learning algorithm is *efficient* if it runs in time polynomial in n, $\frac{1}{\epsilon}$ and $\log \frac{1}{\delta}$.

2.2 Restrictions

We consider the domain $\mathcal{X} = \{0,1\}^n$. A vector $\mathbf{x} = (x_1, \cdots, x_n) \in \mathcal{X}$ is an assignment to the n variables $x_1, \cdots, x_n$. For $1 \leq i \leq n$ we define:

$$\mathbf{x}^i \stackrel{\text{def}}{=} (x_1, \cdots x_{i-1}, x_{i+1}, \cdots, x_n)$$

For $b \in \{0,1\}$ we define:

$$\mathbf{x}_{i \leftarrow b} \stackrel{\text{def}}{=} (x_1, \cdots x_{i-1}, b, x_{i+1}, \cdots, x_n).$$

Let D be a probability distribution over the n dimensional Boolean domain $\mathcal{X}$. Let f be a Boolean function $f : \mathcal{X} \to \{0,1\}$. Restricting a function f by fixing the value of a variable x_i to $b \in \{0,1\}$, is denoted by

$$f_{i \leftarrow b}(\mathbf{x}^i) \stackrel{\text{def}}{=} f(\mathbf{x}_{i \leftarrow b}) = f(x_1, \cdots x_{i-1}, b, x_{i+1}, \cdots, x_n).$$

When we restrict a probability distribution we need also to normalize the sum of the probabilities to one:

$$D_{i \leftarrow b}(\mathbf{x}^i) \stackrel{\text{def}}{=} D(\mathbf{x}^i | x_i = b) = \frac{D(\mathbf{x}_{i \leftarrow b})}{Pr_D(x_i = b)}$$

The following lemma is a simple use of Bayes formula.

Lemma 1 *Let $f(\mathbf{x})$ be a boolean function $f : \{0,1\}^n \to \{0,1\}$, and let $D(\mathbf{x})$ be a probability distribution over $\{0,1\}^n$, then*

$$E_{D_{i\leftarrow b}}[f_{i\leftarrow b}(\mathbf{x}^i)] = \frac{E_D[I_b(x_i)f(\mathbf{x})]}{E_D[I_b(x_i)]}$$

Where $I_b(x)$ is the indicator function ($I_b(x) = 1$ iff $x = b$, otherwise $I_b(x) = 0$).

we define the influence of a variable x_i on the boolean function f as the probability that filpping the variable's value changes the output of f,

$$\mathrm{Inf}_D\,(x_i, f) \stackrel{def}{=} E_{D_{i\leftarrow 1}}[f_{i\leftarrow 1}(\mathbf{x}^i)] - E_{D_{i\leftarrow 1}}[f_{i\leftarrow 0}(\mathbf{x}^i)].$$

2.3 Read-once DNF

A literal ℓ_i is a positive or negative appearance of a variable, e.g. x_i or $\overline{x_i}$. A term m is a conjunction of literals, e.g. $m = x_6 \wedge \overline{x_8} \wedge x_3$. A DNF is a disjunction of terms, i.e.

$$g(\mathbf{x}) = m_1 \vee m_2 \vee \cdots \vee m_s$$

A DNF is *read-once* if each variable appears in at most one term. Let $g(\mathbf{x})$ be a read-once DNF, and let m be the term in which x_i appears. We can rewrite m as $m = g_{i1}(\mathbf{x}^i)\ell_i$ where $g_{i1}(\mathbf{x}^i)$ are the other literals in m. Let $g_{i0}(\mathbf{x}^i)$ be the disjunction of all the terms of $g(\mathbf{x})$ except m, then

$$g(\mathbf{x}) \equiv g_{i0}(\mathbf{x}^i) \vee g_{i1}(\mathbf{x}^i)\ell_i.$$

Let $b_i = 0$ if $\ell_i = x_i$ and $b_i = 1$ if $\ell_i = \overline{x_i}$, then $g_{i\leftarrow b_i} \equiv g_{i0}$ and $g_{i\leftarrow \overline{b_i}} \equiv g_{i0} \vee g_{i1}$.

2.4 Learning using statistical queries

The *Statistical-Query* learning model [Kea93] uses an expectation oracle $STAT$ that has as input a function $SQ(\mathbf{x}, b)$ which is a function of $x \in \mathcal{X}$ and the value of the target concept $b = c(\mathbf{x})$. Note that unlike the standard Statistical Query model, we are using here a function rather than a predicate. In order to maintain the simulation in the PAC model we need to require that the range of SQ is bounded by some constant L, i.e.,

$$SQ : \mathcal{X} \times \{0,1\} \to [0, L].$$

Like in the PAC model we assume a probability distribution $D(\mathbf{x})$ over the domain $\mathcal{X}$. The oracle $STAT$ also receives an accuracy parameter $\mu > 0$. For every input (SQ, μ) the oracle $STAT$ outputs a value $\widehat{E}_{SQ}$ which is μ-close to the true expectation $E_{SQ} = E_D[SQ(\mathbf{x}, c)]$, i.e.,

$$|\widehat{E}_{SQ} - E_{SQ}| \le \mu$$

This implies that $O(\log \frac{L}{\mu})$ bits are sufficient to encode $STAT$'s output.

An algorithm learns *using statistical queries* the concept class $\mathcal{C}$ over distribution D if for any $c \in \mathcal{C}$, given $\epsilon > 0$ and access to an oracle $STAT$, the algorithm outputs a hypothesis $h \in \mathcal{C}$ such that $Error_D(h, c) < \epsilon$.

We say that such a statistical query algorithm is *efficient* if every function SQ given to $STAT$ can be evaluated in time polynomial in n and the algorithm runs in time polynomial in n and $\frac{1}{\epsilon}$ (the running time includes the number of statistical queries the algorithm performs), and the query accuracy μ is also bounded by an inverse of a polynomial in those parameters.

It can be shown [Kea93] that if a concept class $\mathcal{C}$ is *efficiently* learnable *using statistical queries* over distribution D, then $\mathcal{C}$ is *efficiently PAC*-learnable over the distribution D (even in the presence of random classification noise).

We say that the statistical query oracle is *Exact* when $\mu = 0$ (in this case the oracle returns the exact expectation). Since an *Exact* oracle is an ideal entity we ignore it's output encoding complexity. We introduce the exact oracle only as a methodological step in developing our algorithm.

3 The Maximum-Entropy Formalism

The entropy measures the uncertainty about the outcome of an experiment, it's value can be interpreted as the average number of bits needed to encode the samples generated by D. The entropy of a distribution D is

$$H(D) \stackrel{\mathrm{def}}{=} -\sum_{\mathbf{x}\in\mathcal{X}} D(\mathbf{x}) \log D(\mathbf{x}) = E_D[-\log D(\mathbf{x})].$$

We would like to impose constraints on the distribution. A constraint determines the probability of a certain event. (Such constraints may be viewed as a prior.) Our goal is to find a distribution that satisfies those constraints while imposing the least "additional constraints". One way of achieving this, assuming such a distribution exists, is choosing the distribution that maximizes the entropy subject to the constraints imposed.

We first examine the form of the Maximum Entropy distribution, satisfying a given set of constraints. This problem has been widely studied, here we briefly present the basic results. (The interested reader may look at [Pap91, SWG85].) We want to determine the probability distribution $D(\mathbf{x})$ subject to the constraints that the expected value of the function $f_i(\mathbf{x})$ is μ_i, i.e.

$$E[f_i(\mathbf{x})] = \mu_i \quad i = 1, \cdots, m$$

Among all distributions satisfying the above constraints, the one with maximum entropy has the form

$$D(\mathbf{x}) = Ae^{-\sum_{i=1}^{m} \lambda_i f_i(\mathbf{x})}$$

where the parameters λ_i are the solution of the system

$$-\frac{\partial}{\partial \lambda_i} \ln Z = \mu_i \quad i = 1, \cdots, m \ , \qquad \textit{and}$$
$$Z(\lambda_1, \cdots, \lambda_m) \stackrel{\text{def}}{=} \sum_{\mathbf{x} \in \mathcal{X}} e^{-\sum_{i=1}^{m} \lambda_i f_i(\mathbf{x})} \ .$$

The normalization factor is $A = \frac{1}{Z}$. In our special case, we are given n expectations, $E(x_i) = \mu_i$, and n^2 correlations, $E(x_i x_j) = \mu_{ij}$. One can show [Pap91] that the distribution suggested by the maximum entropy method (denoted MEM) has the form

$$D(\mathbf{x}) = A e^{-\mathbf{x}^t \mathbf{M} \mathbf{x} - \mathbf{w}^t \mathbf{x}}$$

Where $\mathbf{M}$ is a symmetric $n \times n$ matrix and $\mathbf{w} \in \Re^n$ is a vector (we denote by $\mathbf{x}^t$ the vector $\mathbf{x}$ transposed).

As one can see the distribution has the form that all the inputs have non-zero probability. If we would like to impose a constraint that some x_i is always one then some of the related λ values might go to infinity (in the case of product distributions λ_i, where $f_i(\mathbf{x}) = x_i$, will go to infinity). A way to deal with this issue is to delete such a variable from the input domain since it is deterministic.

As stated in the introduction, the related computational problem is hard since testing the consistency of the given expectations (existence of any distribution satisfying those constraints) is known to be NP-Complete [KN95, KM93].

4 Properties of MEM distributions

In this section we derive a few properties of MEM distributions. One basic property which will be very important for us is relating the probabilities when a variable is fixed to one or zero. We show that for MEM distributions those probabilities are fairly related. (Note that in a product distribution those probabilities are identical.)

Claim 2 *Let D be a MEM distribution with parameters M and w. For any i, $1 \le i \le n$, there exists a vector $\mathbf{z}_i$, and a scalar r_i such that*

$$D(\mathbf{x}_{i \leftarrow 1}) = e^{-\mathbf{z}_i \mathbf{x}^i - r_i} D(\mathbf{x}_{i \leftarrow 0})$$

where $\mathbf{z}_i = 2M_i^i$ and $r_i = m_{ii} + w_i$. (Recall that M_i^i is the i'th row of the matrix $\mathbf{M}$ with the i'th element deleted).

Proof: Without loss of generality, and to simplify the notation we prove the claim for $i = 1$. We write the symmetric matrix $M = \begin{pmatrix} a & \mathbf{b}^t \\ \mathbf{b} & \mathbf{M}_{11} \end{pmatrix}$, we have

$$\begin{aligned} \ln D(1\,\mathbf{x}^1) &= \ln A - \begin{pmatrix} 1 & \mathbf{x}^1 \end{pmatrix} \begin{pmatrix} a & \mathbf{b}^t \\ \mathbf{b} & \mathbf{M}_{11} \end{pmatrix} \begin{pmatrix} 1 \\ \mathbf{x}^1 \end{pmatrix} \\ &\quad - \begin{pmatrix} 1 & \mathbf{x}^1 \end{pmatrix} \begin{pmatrix} w_1 \\ \mathbf{w}^1 \end{pmatrix} \\ &= \ln A - (\mathbf{x}^1)^t \mathbf{M}_{11} \mathbf{x}^1 - (\mathbf{x}^1)^t \mathbf{w}^1 \\ &\quad - a - 2\mathbf{b}^t \mathbf{x}^1 - w_1 \\ &= \ln D(0\,\mathbf{x}^1) - a - 2\mathbf{b}^t \mathbf{x}^1 - w_1. \end{aligned}$$

Therefore, $D(\mathbf{x}_{1\leftarrow 1}) = e^{-2\mathbf{b}^t \mathbf{x}^1 - (a + w_1)} D(\mathbf{x}_{1 \leftarrow 0})$ ■

A simple conclusion is the following relation for the conditional distributions

$$D_{i \leftarrow 1}(\mathbf{x}^i) = \beta_i e^{-\mathbf{z}_i \mathbf{x}^i - r_i} D_{i \leftarrow 0}(\mathbf{x}^i),$$
$$\textit{where} \qquad \beta_i \stackrel{\text{def}}{=} \frac{Pr_D(x_i = 0)}{Pr_D(x_i = 1)}$$

The above property allows us to compute the expectation of the restricted function $f_{i \leftarrow 0}(\mathbf{x}^i)$ over probability distribution $D_{i \leftarrow 1}(\mathbf{x}^i)$. (Examples of $f_{i \leftarrow 0}(\mathbf{x}^i)$ are only available over the conditional distribution $D_{i \leftarrow 0}(\mathbf{x}^i)$!). The following lemma shows how this is done.

Lemma 3 *Let D be a MEM distribution with parameters M and w, and $f(\mathbf{x})$ be a boolean function. Then*

$$E_{D_{i \leftarrow 1}}[f_{i \leftarrow 0}(\mathbf{x}^i)] = E_{D_{i \leftarrow 0}}[\beta_i e^{-\mathbf{z}_i \mathbf{x}^i - r_i} f_{i \leftarrow 0}(\mathbf{x}^i)],$$

where $\mathbf{z}_i = 2M_i^i$ and $r_i = m_{ii} + w_i$.

We thus have a method to compute the influence $\mathrm{Inf}_D\,(x_i, f)$ of a variable x_i on a boolean function f over a given MEM distribution D.

5 Learning using Exact Statistical Queries oracle

In this section we develop a statistical query algorithm for the special case where we receive the exact expectation each time we query STAT. We use the exact oracle to highlight some of our ideas, and in Section 7 we show how to adapt it to the case where STAT returns the expectation with an additive error.

The following theorem derives a simple condition which enables to learn read-once DNF over MEM distributions using Exact Statistical Queries oracle (i.e. $\mu = 0$ in the SQ Learning model).

Theorem 4 *Let $P(\mathbf{x})$ be a probability distribution over $\{0,1\}^{n-1}$ such that for every $\mathbf{z} \in \{0,1\}^{n-1}$, $P(\mathbf{z}) > 0$. Let $g(\mathbf{x})$ be read-once DNF, The following holds*

1. $E_P[g_{i \leftarrow 1}(\mathbf{x}^i)] > E_P[g_{i \leftarrow 0}(\mathbf{x}^i)]$ *iff* *x_i appears in $g(\mathbf{x})$*
2. $E_P[g_{i \leftarrow 1}(\mathbf{x}^i)] < E_P[g_{i \leftarrow 0}(\mathbf{x}^i)]$ *iff* *$\overline{x_i}$ appears in $g(\mathbf{x})$*
3. $E_P[g_{i \leftarrow 1}(\mathbf{x}^i)] = E_P[g_{i \leftarrow 0}(\mathbf{x}^i)]$ *iff* *$g(\mathbf{x})$ does not depend on x_i*

Proof: If $g(\mathbf{x})$ does not depend on x_i we have $g_{i \leftarrow 1} \equiv g_{i \leftarrow 0}$, and therefore the expectations are identical.

Recall the representation $g(\mathbf{x}) \equiv g_{i0}(\mathbf{x}^i) \vee g_{i1}(\mathbf{x}^i)\ell_i$. If x_i appears in $g(\mathbf{x})$ then $g_{i\leftarrow 1}(\mathbf{x}^i) \equiv g_{i0}(\mathbf{x}^i) \vee g_{i1}(\mathbf{x}^i) \geq g_{i0}(\mathbf{x}^i) \equiv g_{i\leftarrow 0}(\mathbf{x}^i)$. Since g depends on x_i, there is some $\mathbf{z} \in \{0,1\}^{n-1}$ such that $g_{i\leftarrow 1}(\mathbf{z}) > g_{i\leftarrow 0}(\mathbf{z})$. The lemma assumes that $P(\mathbf{z}) > 0$ for any $\mathbf{z}$, therefore, $E_P[g_{i\leftarrow 1}(\mathbf{x}^i)] > E_P[g_{i\leftarrow 0}(\mathbf{x}^i)]$. The case of $\overline{x_i}$ is similar. ■

For a MEM distribution D, $D(\mathbf{x}) > 0$ for every $\mathbf{x}$ (since we will delete all the deterministic variables). Theorem 4 implies that knowing the influences

$$\gamma_i = E_{D_{i\leftarrow 1}}[g_{i\leftarrow 1}(\mathbf{x}^i)] - E_{D_{i\leftarrow 1}}[g_{i\leftarrow 0}(\mathbf{x}^i)] \stackrel{def}{=} \mathrm{Inf}_D\,(x_i, g)$$

is sufficient to decide which literals appear in a read-once DNF. The algorithm itself is very similar to the one for learning read-once DNF under a uniform/product distribution [KLV94]. We first identify the variables on which the target function depends, and then test every two such variables if they appear in the same term. We defer the proofs (which are the statistical query algorithm) to the Appendix. The following lemma shows that we can compute γ_i.

Lemma 5 *Given the parameters* $\mathbf{M}$ *and* $\mathbf{w}$ *of a MEM distribution* D, γ_i *can be computed using an exact statistical queries oracle.*

This allows us to find the variables appearing in a read-once DNF $g(\mathbf{x})$ and their sign, under a given MEM probability distribution $D(\mathbf{x})$. Let S be this set of literals. The next step is to find the terms forming $g(\mathbf{x})$. To do that we have to test for every pair of literals in L whether they appear in the same term or not. The following lemma allows us to use the same method as before to test this.

Lemma 6 *Let* $\ell_i, \ell_j \in S$. *The literals* ℓ_i *and* ℓ_j *appear in the same term of a read-once DNF* $g(\mathbf{x})$ *if and only if* $\mathrm{Inf}_{D_{i\leftarrow b_i}}(x_j, g_{i\leftarrow b_i}) = 0$ *(constraining* ℓ_i *to zero makes* g *not depend on* ℓ_j*). (Recall from Section 2.3 that* $b_i = 0$ *if* $\ell_i = x_i$ *and* $b_i = 1$ *if* $\ell_i = \overline{x_i}$*.)*

Assume that $\ell_i, \ell_j \in S$. In order to test if ℓ_i and ℓ_j appear in the same term of $g(\mathbf{x})$ we check if the read-once DNF $g_{i\leftarrow b_i}(\mathbf{x}^i)$ over probability distribution $D_{i\leftarrow b_i}(\mathbf{x}^i)$ depends on x_j. For this it is sufficient to compute

$$\begin{aligned}\gamma_{i,j} &= E_{D_{i\leftarrow b_i, j\leftarrow \overline{b_j}}}[g_{i\leftarrow b_i, j\leftarrow \overline{b_j}}(\mathbf{x}^{ij})] \\ &\quad - E_{D_{i\leftarrow b_i, j\leftarrow \overline{b_j}}}[g_{i\leftarrow b_i, j\leftarrow b_j}(\mathbf{x}^{ij})] \\ &\stackrel{def}{=} \mathrm{Inf}_{D_{i\leftarrow b_i}}(x_j, g_{i\leftarrow b_i})\,.\end{aligned}$$

Lemma 7 *Given the parameters* $\mathbf{M}$ *and* $\mathbf{w}$ *of a MEM distribution* D, $\gamma_{i,j}$ *can be computed using an exact statistical queries oracle.*

6 Bounded MEM distributions

Recall the general form of a MEM distribution $D(\mathbf{x}) = Ae^{-\mathbf{x}^t\mathbf{M}\mathbf{x}-\mathbf{w}^t\mathbf{x}}$. We call a MEM distribution α-bounded if the inner product of any row of M with every possible $\mathbf{x} \in \mathcal{X}$ is bounded by a constant α, in addition we require that the elements of $\mathbf{w}$ are bounded by α. I.e., for all $\mathbf{x}$ and i

$$|\mathbf{M}_i\mathbf{x}| \leq \alpha \quad \text{and} \quad |w_i| \leq \alpha$$

As we shall see, the value of α is a measure for the relative influence of a variable on the other variables. For example, for a product distribution to be α-bounded implies that the probability of each variable to be 1 is bounded away from both one and zero. The relationship between the value of α and the properties of the related MEM distributions is left for future research.

Unlike product distributions, in MEM distributions the value of one variable influences the distribution of the other variables. Still, we prove that this influence is limited.

Lemma 8 *Let* $D(\mathbf{x})$ *be* α*-bounded MEM distribution. Let* $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$ *differ only in one place, then*

$$|\ln\frac{D(\mathbf{x})}{D(\mathbf{y})}| \leq 3\alpha.$$

Proof: Without loss of generality, we can assume that $\mathbf{x}$ and $\mathbf{y}$ differ in the first place, i.e.

$$\mathbf{x} = (1\,\mathbf{z}) \quad \textit{and} \quad \mathbf{y} = (0\,\mathbf{z})$$

Write the matrix $M = \begin{pmatrix} a & \mathbf{b}^t \\ \mathbf{b} & \mathbf{M}_1 \end{pmatrix}$, and the vector $w = (w_1\mathbf{w}^1)$. By claim 2

$$\ln D(\mathbf{x}) = \ln D(\mathbf{y}) - a - 2\mathbf{b}^t\mathbf{z} - w_1$$

Therefore,

$$|\ln\frac{D(\mathbf{x})}{D(\mathbf{y})}| = |a+2\mathbf{b}^t\mathbf{z}+w_1| \leq |a+\mathbf{b}^t\mathbf{z}|+|\mathbf{b}^t\mathbf{z}|+|w_1| \leq 3\alpha$$

The last inequality follows from the fact that D is an α-bounded MEM distribution. ■

Building on Lemma 8 we show that two vectors that are close in *Hamming* distance, have somewhat similar probabilities.

Lemma 9 *Let* $d(\mathbf{x}, \mathbf{y})$ *be the number of bits in which* $\mathbf{x}$ *and* $\mathbf{y}$ *differ, then,*

$$|\ln\frac{D(\mathbf{x})}{D(\mathbf{y})}| \leq 3\alpha d(\mathbf{x}, \mathbf{y}).$$

Proof: Form a series of vectors, $\mathbf{z}_0 = \mathbf{x}$, $\mathbf{z}_1, \cdots, \mathbf{z}_{d-1}$, $\mathbf{z}_d = \mathbf{y}$, such that $d = d(\mathbf{x}, \mathbf{y})$ and $d(\mathbf{z}_i, \mathbf{z}_{i+1}) = 1$, for $i = 0, \cdots, d-1$. It follows that

$$\begin{aligned} |\ln \frac{D(\mathbf{x})}{D(\mathbf{y})}| &= |\ln D(\mathbf{x}) - \ln D(\mathbf{y})| \\ &= |\sum_{i=0}^{d-1} \ln D(\mathbf{z}_i) - \ln D(\mathbf{z}_{i+1})| \\ &\leq \sum_{i=0}^{d-1} |\ln D(\mathbf{z}_i) - \ln D(\mathbf{z}_{i+1})| \\ &= \sum_{i=0}^{d-1} |\ln \frac{D(\mathbf{z}_i)}{D(\mathbf{z}_{i+1})}| \end{aligned}$$

Now, by Lemma 8,

$$|\ln \frac{D(\mathbf{x})}{D(\mathbf{y})}| \leq \sum_{i=0}^{d-1} |\ln \frac{D(\mathbf{z}_i)}{D(\mathbf{z}_{i+1})}| \leq 3\alpha d.$$

■

A case which will be of special interest to us is when the distance between $\mathbf{x}$ and $\mathbf{y}$ is logarithmic.

Corollary 10 *For a given α bounded MEM distribution D, if $d(\mathbf{x}, \mathbf{y}) \leq c \ln \frac{n}{\epsilon}$ then*

$$(n/\epsilon)^{-3c\alpha} \leq \frac{D(\mathbf{x})}{D(\mathbf{y})} \leq (n/\epsilon)^{3c\alpha}.$$

We show now that for α-bounded MEM distributions long terms are insignificant. First we prove a general Lemma.

Lemma 11 *Let $f(\mathbf{x})$ depend only on $x_{l+1} \cdots x_n$ and $D(\mathbf{x})$ be an α-bounded MEM distribution. Let v be an event forcing the first l inputs of $\mathbf{x}$ to a fixed value, i.e. $x_i = b_i$ for $i = 1 \cdots l$, then*

$$\frac{Pr_D(f)}{(1+e^{3\alpha})^l} \leq Pr_D(f \wedge v) \leq \frac{Pr_D(f)}{(1+e^{-3\alpha})^l}.$$

Proof: Devide the elements of $\{0,1\}^n$ into disjoint subsets, such that all the elements of a subset have the same value of $x_{l+1} \cdots x_n$. Note that $f()$ has the same value on all the 2^l elements of each subset S. Denote by S_T the set of subsets whose elements satisfy $f()$. Each subset $S \in S_T$ has a single element $\mathbf{x}_s$ satisfying v. For every $i = 0 \cdots l$ there are in S exactly $\binom{l}{i}$ elements y_i that differ from x_s in exactly i positions. By Lemma 9, the probability of such a y_i is at least $(e^{-3\alpha})^i D(\mathbf{x}_s)$ and at most $(e^{3\alpha})^i D(\mathbf{x}_s)$. Since the total probability of all the elements in S_T is $Pr_D(f)$, we have

$$\begin{aligned} Pr_D(f) &= \sum_{S \in S_T} \sum_{y \in S} D(y) \geq \sum_{S \in S_T} \sum_{i=0}^{l} \binom{l}{i} (e^{-3\alpha})^i D(\mathbf{x}_s) \\ &= (1+e^{-3\alpha})^l \sum_{S \in S_T} D(\mathbf{x}_s) \\ &= (1+e^{-3\alpha})^l Pr_D(f \wedge v), \end{aligned}$$

The other direction is simmilar, and the lemma follows. ■

Corollary 12 *Let $g(\mathbf{x})$ be a DNF and $D(\mathbf{x})$ be an α-bounded MEM distribution. Let m be a term of g with k literals, then $Pr_D(m=1) \leq \frac{1}{(1+e^{-3\alpha})^k}$.*

We use Corollary 12 to prove that the learning algorithm can afford to ignore long terms.

Theorem 13 *Let $g(\mathbf{x})$ be a DNF and $D(\mathbf{x})$ be an α-bounded MEM distribution. Denote by $g_{\alpha,\epsilon}$ the disjunction of all the terms of g of length at most $k = e^{3\alpha} \ln \frac{n}{\epsilon}$. Then $Error_D(g, g_{\alpha,\epsilon}) < \epsilon$.*

Proof: Let m be a term of length more than $k = e^{3\alpha} \ln \frac{n}{\epsilon}$. By Corollary 12 we have that $Pr_D(m=1) < \frac{\epsilon}{n}$. Therefore,

$$\begin{aligned} Error_D(g, g_{\alpha,\epsilon}) &= Pr_D(g = 1 \wedge g_{\alpha,\epsilon} = 0) \leq \\ &\leq \sum_{m \in g, m \notin g_{\alpha,\epsilon}} Pr_D(m=1) < \epsilon. \end{aligned}$$

■

7 Learning using Statistical Queries

The existence of exact statistical query oracle is not a realistic assumption. However, relaxing the requirement to approximate the expectation makes it possible to simulate such an oracle given examples of the target function. We show here how *Statistical Queries* (with an accuracy parameter μ) can be used to efficiently learn read-once DNF over α-bounded MEM distributions. We first concentrate on the special case where all the terms are of size at most $O(\ln n/\epsilon)$, later we show how to extend this to the general case.

The statistical query oracle STAT returns a value which is at most μ away from the real expectation. Theorem 4 states that the target function g depends on a variable x_i if and only if $\gamma_i = E_{D_{i\leftarrow 1}}[g_{i\leftarrow 1}] - E_{D_{i\leftarrow 1}}[g_{i\leftarrow 0}] \neq 0$, and that the sign of that γ_i determines the sign of the variable x_i in g. Finding a lower bound on the value γ_i of the short terms is the main task of this section. Once we establish a lower bound Γ, it is sufficient to run the statistical queries of the algorithm (presented in Lemma 5 and Lemma 7 and with the choice of appropriate accuracy parameters), such that the total error is less than $\Gamma/2$. This will allow us to recover the true sign of γ_i from the values that STAT returns. The main concern is to obtain a bound Γ which is polynomial in n and $1/\epsilon$, since this is a requirement in order for the statistical query algorithm to be efficient.

In the analysis we can assume that the target function g satisfies $E_D(\overline{g}) > \epsilon/2$. (Otherwise we can estimate g by $h \equiv 1$). Recall the notation $g = \ell_i g_{i1} \vee g_{i0}$. We denote the term in which ℓ_i appears by m_i ($m_i \equiv \ell_i g_{i1}$).

Lemma 14 *Let D be an α-bounded MEM distribution and $g(\mathbf{x})$ be a read-once DNF. If $E_D(\overline{g}) > \frac{\epsilon}{2}$ and m_i is a term of size at most $l \leq c \ln \frac{n}{\epsilon}$ then $E_D(m_i \wedge \overline{g_{i0}}) > \frac{\epsilon}{2}(\frac{\epsilon}{n})^{c(3\alpha+1)}$*

Proof: First we bound $E_D(\overline{g_{i0}}) > E_D(\overline{g_{i0}} \wedge \overline{m_i}) = E_D(\overline{g}) > \epsilon/2$. We can use Lemma 11, since m_i and g_{i0} depend on different variables. By Lemma 11 we have

$$E_D(m_i \wedge \overline{g_{i0}}) > \frac{\epsilon/2}{(1+e^{3\alpha})^l} \geq \frac{\epsilon}{2}(\frac{\epsilon}{n})^{c(3\alpha+1)}$$

and the lemma follows. ■

¿From the above lemma we can deduce the following corollary on the value of Γ.

Corollary 15 *Let D be an α-bounded MEM distribution and Let $g(\mathbf{x})$ be a read-once DNF such that $E_D(\overline{g}) > \frac{\epsilon}{2}$. If ℓ_i appears in a term of size at most $c \ln \frac{n}{\epsilon}$ then*

$$\begin{aligned} |\gamma_i| &= |E_{D_{i\leftarrow 1}}(g_{i\leftarrow 1}) - E_{D_{i\leftarrow 1}}(g_{i\leftarrow 0})| \\ &> (\epsilon/2)(\epsilon/n)^{c(3\alpha+1)} = \Gamma \end{aligned}$$

Proof: We assume $\ell_i = x_1$ (The other case is simmilar).

$$\begin{aligned} &E_{D_{i\leftarrow 1}}[g_{i\leftarrow 1}] - E_{D_{i\leftarrow 1}}[g_{i\leftarrow 0}] = E_{D_{i\leftarrow 1}}[g_{i\leftarrow 1} - g_{i\leftarrow 0}] \\ &\quad = E_{D_{i\leftarrow 1}}[g_{i0} \vee g_{i1} - g_{i0}] = E_{D_{i\leftarrow 1}}[\overline{g_{i0}} \wedge g_{i1}] \\ &\quad = \frac{E_D[\overline{g_{i0}} \wedge g_{i1} \wedge x_i]}{Pr_D(x_i)} \geq E_D[\overline{g_{i0}} \wedge m_i] \\ &\quad > (\epsilon/2)(\epsilon/n)^{c(3\alpha+1)} \end{aligned}$$

■

Once the significant variables are found we have to test if pairs of literals belong to the same term. We can derive a similar bound on the true expectations in this case as well. (This is because if D is α-bounded then so is $D_{i\leftarrow b}$, and if the terms of g are of size at most $c \log \frac{n}{\epsilon}$ so are the terms of $g_{i\leftarrow b}$). This establish the learnability result for read-once DNF with short terms.

Theorem 16 *The concept class read-once DNF with terms of size $O(\log n/\epsilon)$ can be efficiently learned with Statistical Queries over a given α-bounded MEM distribution.*

In what follows we sketch the modifications in order to handle an arbitrary read-once DNF. The main worry is that the presence of long terms will hurt the learning process. First note that using Corollary 15 we can assure to recover all the terms of size at most $c \ln \frac{n}{\epsilon}$, by definition this is $g_{\alpha,\epsilon}$. Since, by Theorem 13, longer terms are insignificant, the short terms are sufficient for a good approximation of the target g.

The problem is that the algorithm might find short terms that are not terms of the target function but may be part of longer terms. Such terms increase the error when they are one and the target function is zero. A special worry are such terms that increase the error significantly, we call such terms harmful.

The idea is to identify and delete the harmful terms. For a given term m, this can be done by making a statistical query on the probability that m is one and the target function is zero. Once we deleted the harmful terms we have a good approximation of the target read-once DNF. In the final version of the paper we show the details of the following theorem.

Theorem 17 *The concept class read-once DNF can be efficiently learned with Statistical Queries over a given α-bounded MEM distribution.*

We conclude with the learning algorithm.

Literal$(i, \Gamma, \mathbf{M}, \mathbf{w})$
Let $z = 2\mathbf{M}_i^i$, $r = m_{ii} + w_i$.
Estimate $\beta = \frac{Pr_D(x_i=0)}{Pr_D(x_i=1)}$
Estimate $\gamma_1 = Pr_{D_{i\leftarrow 1}}[g_{i\leftarrow 1}]$
Estimate $\gamma_0 = Pr_{D_{i\leftarrow 0}}[\beta e^{-\mathbf{z}\mathbf{x}^i - r} g_{i\leftarrow 0}]$.
IF $|\gamma_1 - \gamma_0| < \frac{\Gamma}{2}$ RETURN 0
ELSE IF $\gamma_1 > \gamma_0$ RETURN x_i
ELSE RETURN $\overline{x_i}$

SameTerm$(i, j, \Gamma, \mathbf{M}, \mathbf{w})$
Let $z = 2\mathbf{M}_j^{ij}$, $r = m_{jj} + w_j$.
Estimate $\beta = \frac{Pr_D(x_i=1)\wedge(x_j=0)}{Pr_D(x_i=1)\wedge(x_j=1)}$.
Estimate $\gamma_1 = Pr_{D_{i\leftarrow 0, j\leftarrow 1}}[g_{i\leftarrow 0, j\leftarrow 1}]$
Estimate $\gamma_0 = Pr_{D_{i\leftarrow 0, j\leftarrow 0}}[\beta e^{-\mathbf{z}\mathbf{x}^{ij} - r} g_{i\leftarrow 0, j\leftarrow 0}]$.
IF $|\gamma_1 - \gamma_0| < \frac{\Gamma}{2}$ RETURN 1
ELSE RETURN 0

LearnMaxEnt$(\epsilon, \mathbf{M}, \mathbf{w}, \alpha)$
Estimate $p \equiv Prob[g = 0]$.
IF $p < \epsilon/2$ OR $p > 1 - \epsilon/2$ THEN DONE.
Let $\Gamma = (\epsilon/2)(\epsilon/n)^{e^{3\alpha}(3\alpha+1)}$.
Let $\mathcal{S} = \emptyset$.
FOR each variable x_i,
  Let $Term_i = \emptyset$.
  Estimate $p_i = Pr_D(x_i = 1)$.
  IF $p_i > \frac{\Gamma}{2}$
    CALL $\ell_i =$ **Literal**$(i, \Gamma, \mathbf{M}, \mathbf{w})$
    IF $\ell_i \neq 0$ $\mathcal{S} = \mathcal{S} \cup \{i\}$.
FOR each $i = 1 \ldots n$ s.t $i \in \mathcal{S}$
  FOR each $j > i$ s.t $j \in \mathcal{S}$
    Estimate $p_{ij} = Pr_D((x_i = 1) \wedge (x_j = 1))$.
    IF $p_{ij} > \frac{\Gamma}{n}$
      IF **SameTerm**$(i, j, \Gamma, \mathbf{M}, \mathbf{w})$
        $Term_i = Term_i \wedge \ell_j$.
        $\mathcal{S} = \mathcal{S} - \{j\}$.
FOR each $i = 1 \ldots n$ s.t $Term_i \neq 0$

Estimate $t_i = Pr_D(Term_i \wedge \overline{g})$.
IF $t_i > \frac{\Gamma}{2}$ THEN $Term_i = 0$.
RETURN $\bigvee Term_i$.

Claim 18 *Assume the estimates are accurate with $\mu = \frac{\Gamma}{2}$ then With probability at least $1 - \frac{\delta}{2n}$, the algorithm recovers in poynomial time all the terms of $g_{\alpha,\epsilon}$, and no harmfull terms.*

In the previous sections we showed how a result about the learnability of read once DNF over product distributions can be extended to α-bounded MEM distributions. We now continue and briefly describe how other results can be extended in a simmilar way.

8 Learning Monotone $k\mu$ DNF

In [HM91] it was shown how to efficiently learn over product distributions the class of monotone DNF where each variable appears in no more than k terms (denoted $k\mu$-DNF). The learning algorithm did not reconstruct the DNF, instead, it efficiently identified all the minimal blocking sets for each variable (a set of variables ρ blocks the variable x_i over a function f iff $f_{\rho\leftarrow 0}$ does not depend on x_i, i.e. iff $\mathrm{Inf}(x_i, f_{\rho\leftarrow 0}) = 0$). It was shown that knowing an approximation of the set of minimal blocking sets for each variable, enables a good approximation of the $k\mu$-DNF.

8.1 Computing using blocking sets

We first present (as in [HM91]) a function that computes a monotone function $f()$ given its blocking sets. The fiction *compute*() is given as input a vector of literals $x_1, \ldots, x_n$ and for each literal a set of minimal blocking sets $\mathcal{B}_1, \ldots \mathcal{B}_n$ where $\mathcal{B}_i \subset 2^{\{1,\cdots,n\}}$. The computed value is $f(x_1, \ldots, x_n)$ as stated by the Lemma that follows.

FUNCTION $compute(x_1, \ldots, x_n, \mathcal{B}_1, \ldots, \mathcal{B}_n)$
IF $\exists x_i = 1$ such that
  $\forall A \in \mathcal{B}_i$ there exists $j \in A$,
    such that $x_j = 1$.
THEN RETURN 1
ELSE RETURN 0.

Lemma 19 *[HM91] Let $f(x_1, \ldots, x_n)$ be a monotone function, and $\mathcal{B}_i$, $1 \leq i \leq n$, is a maximal collection of minimal blocking sets of the variable x_i. Then,*

$$compute(x_1, \ldots, x_n, \mathcal{B}_1, \ldots, \mathcal{B}_n) = f(x_1, \ldots, x_n)$$

8.2 Identifying blocking sets

By the definition, blocking sets ρ of a variable x_i can be identifyed by measuring the influence of a variable x_i on $g_{\rho\leftarrow 0}$. We will prove that over α-bounded MEM distributions, for an approximation $g_{\alpha,\epsilon}$ of the kμ DNF g, a lower bound on that influence for non-blocking sets is polynomial in n and $\frac{1}{\epsilon}$. This will allow us to eficiently identify the blocking sets. This generalizes the result of [HM91].

Lemma 20 *Let D be an α-bounded MEM distribution and*

$$g = T_0 \vee T_1 \vee T_2 \vee \cdots \vee T_s \vee R_1 \vee R_2 \vee \cdots \vee R_q$$

be a monotone $k\mu$ DNF such that T_0 is a term of size l and $T_1 \cdots T_s$ are the terms sharing literals with T_0. If $Pr_D(\overline{g}) > \frac{\epsilon}{2}$ then

$$Pr_D(T_0 \wedge \overline{T_1} \wedge \cdots \wedge \overline{T_s} \wedge \overline{R_1} \wedge \cdots \wedge \overline{R_q}) > \frac{\epsilon/2}{(1+e^{3\alpha})^{kl}}$$

Proof: Without loss of generallity we can assume that $T_0 = x_1 \wedge \cdots \wedge x_l$. By the monotony of g, every term T_i, $i = 1 \cdots s$, has at least one literal not appearing in T_0. Denote by $S = \{x_{j_1} \cdots x_{j_s}\}$ a set containing one literal from every T_i, $i = 1 \cdots s$, and none from T_0. Since g is $k\mu$ we have $s \leq (k-1)l$. Denote by $R_{a_1} \cdots R_{a_t}$ the terms among $R_1 \cdots R_q$ with no literal in S. First we bound

$$Pr_D(\overline{R_{a_1}} \wedge \cdots \wedge \overline{R_{a_t}}) > Pr_D(\overline{g}) > \epsilon/2$$

Now, using the above definitions we write

$$\begin{aligned} &Pr_D(T_0 \wedge \overline{T_1} \wedge \cdots \wedge \overline{T_s} \wedge \overline{R_1} \wedge \cdots \wedge \overline{R_q}) \\ &\quad > Pr_D(T_0 \wedge \overline{x_{j_1}} \wedge \cdots \wedge \overline{x_{j_s}} \wedge \overline{R_1} \wedge \cdots \wedge \overline{R_q}) \\ &\quad = Pr_D(T_0 \wedge \overline{x_{j_1}} \wedge \cdots \wedge \overline{x_{j_s}} \wedge \overline{R_{a_1}} \wedge \cdots \wedge \overline{R_{a_t}}) \end{aligned}$$

We can now use Lemma 11, this is becouse $T_0 \wedge \overline{x_{j_1}} \wedge \cdots \wedge \overline{x_{j_s}}$ impose no more than $l + s \leq kl$ constraints on variables not belonging to any of the terms $R_{a_1} \cdots R_{a_t}$. We bound further

$$Pr_D(T_0 \wedge \overline{x_{j_1}} \wedge \cdots \wedge \overline{x_{j_s}} \wedge \overline{R_{a_1}} \wedge \cdots \wedge \overline{R_{a_t}}) \geq \frac{\epsilon/2}{(1+e^{3\alpha})^{kl}} \quad ,$$

Which proves the lemma. ■

Note that for the case where T_0 is a term of size at most $l = c \ln \frac{n}{\epsilon}$, the Lemma generalizes Lemma 14 for monotone $k\mu$ DNF. (The resulting lower bound becomes $\frac{\epsilon}{2}(\frac{\epsilon}{n})^{kc(3\alpha+1)}$.)

Lemma 21 *Let D be an α-bounded MEM distribution and g be a monotone $k\mu$ DNF such that $Pr_D(\overline{g}) > \frac{\epsilon}{2}$. If ρ is not a blocking set for x_i then*

$$\mathrm{Inf}_{D_{\rho\leftarrow 0}}(x_i, g_{\rho\leftarrow 0}) \geq \frac{\epsilon/2}{(1+e^{3\alpha})^{kl}} \quad ,$$

where l is the size of the smallest term that includes x_i in $g_{\rho\leftarrow 0}$.

Proof: Let T_0 be the term of size l that includes x_i and no variable in ρ. We can write

$$\begin{aligned} g &= T_0 \vee T_1 \vee \cdots \vee T_s \vee R_1 \vee \cdots \vee R_q \\ g_{\rho\leftarrow 0} &= T_0 \vee T_{j_1} \vee \cdots \vee T_{j_v} \vee R_{a_1} \vee \cdots \vee R_{a_t} \end{aligned}$$

Where $T_1 \cdots T_s$ are the terms sharing variables with T_0. Without loss of generality we can assume that $T_{j_1} \cdots T_{j_u}$ are the terms of $g_{\rho\leftarrow 0}$ in which x_i appears. We have,

$$\begin{aligned} &\mathrm{Inf}_{D_{\rho\leftarrow 0}}(x_i, g_{\rho\leftarrow 0}) \\ &\stackrel{def}{=} Pr_{D_{\rho\leftarrow 0, i\leftarrow 1}}(g_{\rho\leftarrow 0, i\leftarrow 1}) - Pr_{D_{\rho\leftarrow 0, i\leftarrow 1}}(g_{\rho\leftarrow 0, i\leftarrow 0}) \\ &= Pr_{D_{\rho\leftarrow 0, i\leftarrow 1}}((T_0 \vee T_{j_1} \vee \cdots \vee T_{j_u}) \wedge \\ &\quad (\overline{T}_{j_{u+1}} \wedge \cdots \wedge \overline{T}_{j_v} \wedge \overline{R}_{a_1} \wedge \cdots \wedge \overline{R}_{a_t})) \\ &\geq Pr_{D_{\rho\leftarrow 0}}(T_0 \wedge \overline{T}_{j_1} \wedge \cdots \wedge \overline{T}_{j_v} \wedge \overline{R}_{a_1} \wedge \cdots \wedge \overline{R}_{a_t}) \end{aligned}$$

Since $D_{\rho\leftarrow 0, i\leftarrow 1}$ is α-bounded MEM we can use Lemma 20, hence,

$$\begin{aligned} &\mathrm{Inf}_{D_{\rho\leftarrow 0}}(x_i, g_{\rho\leftarrow 0}) \\ &\geq Pr_{D_{\rho\leftarrow 0}}(T_0 \wedge \overline{T}_{j_1} \wedge \cdots \wedge \overline{T}_{j_v} \wedge \overline{R}_{a_1} \wedge \cdots \wedge \overline{R}_{a_t}) \\ &\geq \frac{\epsilon/2}{(1+e^{3\alpha})^{kl}} \end{aligned}$$

■

8.3 ϵ-blocking sets

We say that a set ρ of variables ϵ-blocks the variable x_i of a function f iff

$$\mathrm{Inf}(x_i, f_{\rho\leftarrow 0}) < \frac{\epsilon}{2}(1+e^{3\alpha})^{-k\lambda} \qquad \textit{where } \lambda = e^{3\alpha} \ln \frac{2kn}{\epsilon}.$$

Note that every blocking set is an ϵ-blocking set, and by Lemma 21, if ρ ϵ-blocks x_i, it contains a variable from every term of length $\leq \lambda$ that contains x_i.

Lemma 22 *Let D be an α-bounded MEM distribution, let g be a $k\mu$ DNF formula, $\mathcal{B}_i$ the set of all ϵ-blocking sets for x_i, and $\mathcal{B} = \mathcal{B}_1, \ldots, \mathcal{B}_n$. Then,*

$$Pr_D(compute(x, \mathcal{B}) \neq g(x)) \leq \epsilon/2,$$

Proof: Similar to [HM91], since every blocking set is an approximate blocking set, $\mathcal{B}_i$ includes all the blocking sets of variable x_i. This implies that if $compute(x, \mathcal{B}) = 1$ then $g(x) = 1$. Therefore the only errors are when $compute(x, \mathcal{B}) = 0$ while the input x satisfies some term(s) of g (i.e. $g(x) = 1$).

As noted before, an approximate blocking set for x must contain a variable from every term containing x, whose length is at most λ. Hence an example satisfying such a term must have a variable set to 1 in each approximate blocking set for x, and therefore will be correctly evaluated by *compute*.

By Corollary 12, the contribution of all g's terms with length more than λ is at most $kn(1+e^{3\alpha})^{-\lambda} = \epsilon/2$, which completes the proof. ■

8.4 The algorithm

The resulting algorithm for identifying ϵ-blocking sets is identical to the one of [HM91].

Let $\lambda = \ln \frac{2kn}{\epsilon}$.
Estimate $p \equiv Prob[g = 0]$.
IF $p < \epsilon/2$ THEN DONE.
FOR each variable x_i,
    Let $\mathcal{B}_i = \emptyset$.
    FOR each set ρ of at most k variables,
        Estimate $\alpha_{\rho,i} \equiv Pr_{D_{\rho\leftarrow 0, i\leftarrow 1}}(g_{\rho\leftarrow 0, i\leftarrow 1})$,
        Estimate $\beta_{\rho,i} \equiv Pr_{D_{\rho\leftarrow 0, i\leftarrow 1}}(g_{\rho\leftarrow 0, i\leftarrow 0})$.
        IF $\alpha_{\rho,i} - \beta_{\rho,i} < p(1+e^{3\alpha})^{-k\lambda-1}$
            THEN $\mathcal{B}_i = \mathcal{B}_i \cup \{\rho\}$.

We conclude with the resulting theorem.

Theorem 23 *Let g be a $k\mu$-DNF. Let D be an α-bounded MEM distribution. Then $g(\mathbf{x})$ can be efficiently approximated.*

References

[AK91] Dana Angluin and Michael Kharitonov. When won't membership queries help? In *Proceedings of STOC '91*, pages 44–454. ACM, 1991.

[BI88] Gyora M. Benedek and Alon Itai. Learnability by fixed distributions. In *Workshop on Computational Learning Theory*, pages 80–90. Morgan Kaufmann, 1988.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* John Wiley and Sons Inc., 1991.

[HKLW90] David Haussler, Michael Kearns, Nick Littlestone, and Manfred K. Warmuth. Equivalence of models for polynomial learnability. *Information and Computation*, 1990.

[HM91] Thomas Hancock and Yishay Mansour. Learning Monotone $k\mu$ DNF Formulas on Product Distributions. In *Proceedings of COLT '91*, pages ?–?. Morgan Kaufmann, 1991.

[Kea93] Michael Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 392 – 352, 1993.

[KM93] D. Koller and N. Megiddo. Constructing small sample spaces satisfying given constraints. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 268–277, 1993.

[KN95] Joe Kilian and Moni Naor. On the complexity of statistical reasoning. In *Proceedings of the 3rd Israel Symposium on Theory* of *Computing and Systems*, pages 209–217, 1995.

[KV88] Michael Kearns and Leslie G. Valiant. Learning boolean formulae or finite automata is as hard as factoring. Technical Report TR 14-88, Harvard University Aiken Computation Laboratory, 1988.

[KLV94] M. Kearns, M. Li and L. Valiant. Learning boolean formulas. *Journal of the ACM*, 41(6):1298–1328, 1994.

[Nat87] B. K. Natarajan. On learning boolean functions. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 296–304, New York, New York, May 1987.

[Pap91] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*, Chapter 15. McGraw-Hill, third edition, 1991.

[PV88] Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.

[SWG85] C.Ray Smith and Jr. W.T. Grandy, editors. *Maximum-Entropy and Bayesian Methods in Inverse Problems.* D.Reidel Publishing Company, 1985.

[Val84] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

A Proof of Lemmas

Proof of Lemma 5: Recall that

$$\gamma_i = E_{D_{i\leftarrow 1}}[g_{i\leftarrow 1}(\mathbf{x}^i)] - E_{D_{i\leftarrow 1}}[g_{i\leftarrow 0}(\mathbf{x}^i)]$$

For the first expectation we form the following queries :

$$SQ_1(\mathbf{x}, b) \stackrel{\text{def}}{=} I_1(x_i)b \quad , \quad SQ_2(\mathbf{x}, b) \stackrel{\text{def}}{=} I_1(x_i)$$

(Recall that $I_b(x_i) = 1$ iff $x_i = 1$). The queries, given to the oracle, yield the exact expectations $E_1 = E_D[I_1(x_i)g(\mathbf{x})]$ and $E_2 = E_D[I_1(x_i)]$ respectively. Now, by Lemma 1 we have

$$E_{D_{i\leftarrow 1}}[g_{i\leftarrow 1}(\mathbf{x}^i)] = \frac{E_1}{E_2}$$

For the second expectation we form the following queries

$$SQ_3(\mathbf{x}, b) \stackrel{\text{def}}{=} I_0(x_i) \quad , \quad SQ_4(\mathbf{x}, b) \stackrel{\text{def}}{=} I_1(x_i)$$

The returned expectations E_3 and E_4 (respectively) are used to compute the transformation factor $\beta_i = \frac{E_3}{E_4}$. Two more queries are needed:

$$SQ_5(\mathbf{x}, b) \stackrel{\text{def}}{=} \beta_i e^{-\mathbf{z}_i \mathbf{x}^i - r_i} I_0(x_i)b \quad , \quad SQ_6(\mathbf{x}, b) \stackrel{\text{def}}{=} I_0(x_i)$$

The queries, given to the oracle, yield the exact expectations $E_5 = E_D[\beta_i e^{-\mathbf{z}_i \mathbf{x}^i - r_i} I_0(x_i)g(\mathbf{x})]$ and $E_6 = E_D[I_0(x_i)]$ respectively. Again, we have

$$E_{D_{i\leftarrow 1}}[g_{i\leftarrow 0}(\mathbf{x}^i)] = E_{D_{i\leftarrow 0}}[\beta_i e^{-\mathbf{z}_i \mathbf{x}^i - r_i} g_{i\leftarrow 0}(\mathbf{x}^i)] = \frac{E_5}{E_6}$$

Where the first equation follows from Lemma 3, and the second from Lemma 1. ■

Proof of Lemma 7: We prove the lemma for the case that $b_i = 0$ and $b_j = 0$, the other cases are similar. We have,

$$\gamma_{i,j} = E_{D_{i\leftarrow 0, j\leftarrow 1}}[g_{i\leftarrow 0, j\leftarrow 1}(\mathbf{x}^{ij})] - E_{D_{i\leftarrow 0, j\leftarrow 1}}[g_{i\leftarrow 0, j\leftarrow 0}(\mathbf{x}^{ij})].$$

For the first expectation we form the following queries :

$$SQ_1(\mathbf{x}, b) \stackrel{\text{def}}{=} I_0(x_i)I_1(x_j)b \quad , \quad SQ_2(\mathbf{x}, b) \stackrel{\text{def}}{=} I_0(x_i)I_1(x_j)$$

The oracle, given the queries, returns the exact expectations $E_1 = E_D[I_0(x_i)I_1(x_j)g(\mathbf{x})]$ and $E_2 = E_D[I_0(x_i)I_1(x_j)]$ respectively. Now, a double implication of Lemma 1 yields

$$E_{D_{i\leftarrow 0, j\leftarrow 1}}[g_{i\leftarrow 0, j\leftarrow 1}(\mathbf{x}^{ij})] = \frac{E_{D_{i\leftarrow 0}}[I_1(x_j)g_{i\leftarrow 0}(\mathbf{x}^i)]}{E_{D_{i\leftarrow 0}}[I_1(x_j)]} = \frac{E_1}{E_2}$$

For the second expectation we form the following queries

$$SQ_3(\mathbf{x}, b) \stackrel{\text{def}}{=} I_0(x_i)I_0(x_j) \quad , \quad SQ_4(\mathbf{x}, b) \stackrel{\text{def}}{=} I_0(x_i)I_1(x_j)$$

The returned expectations E_3 and E_4 (respectively) are used to compute the transformation factor $\beta_{ij} = \frac{E_3}{E_4}$. Two more queries are needed:

$$\begin{aligned} SQ_5(\mathbf{x}, g) &\stackrel{\text{def}}{=} \beta_{ij} e^{-\mathbf{z}_{ij}\mathbf{x}^{ij} - r_{ij}} I_0(x_i)I_0(x_j)g \\ SQ_6(\mathbf{x}, g) &\stackrel{\text{def}}{=} I_0(x_i)I_0(x_j) \end{aligned}$$

Where $\mathbf{z}_{ij}$ and r_{ij} are recursively defined as in Claim 2. The queries, given to the oracle, yield the exact expectations $E_5 = E_D[\beta_{ij} e^{-\mathbf{z}_{ij}\mathbf{x}^{ij} - r_{ij}} I_0(x_i)I_0(x_j)g(\mathbf{x})]$ and $E_6 = E_D[I_0(x_i)I_0(x_j)]$ respectively. Again, we have

$$\begin{aligned} &E_{D_{i\leftarrow 0, j\leftarrow 1}}[g_{i\leftarrow 0, j\leftarrow 0}(\mathbf{x}^{ij})] \\ &\quad = E_{D_{i\leftarrow 0, j\leftarrow 0}}[\beta_{ij} e^{-\mathbf{z}_{ij}\mathbf{x}^{ij} - r_{ij}} g_{i\leftarrow 0, j\leftarrow 0}(\mathbf{x}^{ij})] \\ &\quad = \frac{E_{D_{i\leftarrow 0}}[I_0(x_j)\beta_{ij} e^{-\mathbf{z}_{ij}\mathbf{x}^{ij} - r_{ij}} g_{i\leftarrow 0}(\mathbf{x}^i)]}{E_{D_{i\leftarrow 0}}[I_0(x_j)]} = \frac{E_5}{E_6} \end{aligned}$$

Where the first equation follows from Lemma 3, and the other two from Lemma 1. ■