Machine Learning, 14, 47–81 (1994)
© 1994 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Learning Probabilistic Read-once Formulas on Product Distributions

ROBERT E. SCHAPIRE SCHAPIRE@RESEARCH.ATT.COM
AT&T Bell Laboratories, 600 Mountain Avenue, Room 2A-424, Murray Hill, NJ 07974

Editors: Ming Li and Leslie Valiant

Abstract. This paper presents a polynomial-time algorithm for inferring a probabilistic generalization of the class of read-once Boolean formulas over the usual basis {AND, OR, NOT}. The algorithm effectively infers a good approximation of the target formula when provided with random examples which are chosen according to any *product distribution*, i.e., any distribution in which the setting of each input bit is chosen independently of the settings of the other bits. Since the class of formulas considered includes ordinary read-once Boolean formulas, our result shows that such formulas are PAC learnable (in the sense of Valiant) against any product distribution (for instance, against the uniform distribution). Further, this class of probabilistic formulas includes read-once formulas whose behavior has been corrupted by large amounts of random noise. Such noise may affect the formula's output ("misclassification noise"), the input bits ("attribute noise"), or it may affect the behavior of individual gates of the formula. Thus, in this setting, we show that read-once formula's can be inferred (approximately), despite large amounts of noise affecting the formula's behavior.

Keywords: computational learning theory, PAC-learning, learning with noise, read-once formulas, product distributions

1. Introduction

This paper describes an algorithm for learning a probabilistic generalization of the class of read-once formulas over the usual basis {AND, OR, NOT}. (A formula is *read-once* if each variable appears at most once in the formula.) This algorithm is based on a statistical technique for discovering the structure of a read-once formula. An especially nice feature of this technique is its powerful resistance to noise or randomness.

Similar to the Valiant (1984) model, we consider the problem of learning from randomly chosen examples. We apply our method to a probabilistic generalization of the class of all read-once Boolean formulas constructed from the usual basis {AND, OR, NOT}. We show that an arbitrarily good approximation of such formulas can be inferred in polynomial time against any *product distribution* (i.e., any distribution in which the setting of each variable is chosen independently of the settings of the other variables). For example, this shows that the class of read-once Boolean formulas over the usual basis can be learned in polynomial time against the uniform distribution in the sense of Valiant. This contrasts sharply with the results of Kearns and Valiant (1989) which show that Boolean formulas, even if read-once, cannot be learned against *arbitrary* distributions.

The problem of learning Boolean formulas against special distributions has been considered by a number of other authors. In particular, our technique closely resembles that used by Goldman, Kearns and Schapire (1990) for exactly identifying certain classes of read-once formulas when the observed examples are chosen randomly according to specific, fixed and simple distributions. This technique also resembles that of Kearns et al. (1987) for learning the class of read-once formulas in disjunctive normal form (DNF) against the uniform distribution. A similar result, though based on a different method, was obtained by Pagallo and Haussler (1989). Our results extend theirs to a much broader class of read-once formulas. Their results were also extended in a different direction by Hancock and Mansour (1991).

Linial, Mansour and Nisan (1989) used a technique based on Fourier spectra to learn the class of constant-depth formulas (constructed from gates of unbounded fan-in) against the uniform distribution. Furst, Jackson and Smith (1991) generalized this result to learn this same class against any product distribution. Verbeurgt (1990) gives a different algorithm for learning DNF-formulas against the uniform distribution. However, all three of these algorithms require quasi-polynomial ($n^{\mathrm{polylog}(n)}$) time, though Verbeurgt's procedure only requires a polynomial-size sample.

The class of read-once Boolean formulas has been considered previously by a number of authors. Angluin, Hellerstein and Karpinski (1993) proved that this class of formulas can be exactly identified using membership and equivalence queries. This result has since been extended to much broader bases by Hancock (1990), Hellerstein and Karpinski (1990), and Hancock and Hellerstein (1991). Particularly relevant is the algorithm given by these latter authors for learning read-once formulas over fields in which the gate operations are addition and multiplication; their algorithm bears very strong resemblance to the one presented in the current paper. Their work was recently extended by Bshouty, Hancock and Hellerstein (1992) to also handle division gates.

We adopt from Kearns and Schapire (1990) the notion of a *probabilistic concept (p-concept).* (This learning model is essentially equivalent to Yamanishi's (1992) model for learning "stochastic rules.") A p-concept c is a function that maps each input-variable assignment x to a real number $c(x)$ between 0 and 1. We interpret $c(x)$ as the *probability* that instance x will be positively classified. Thus, in the p-concept model, a randomly labeled example is chosen as follows: first, an instance x is chosen at random according to the target distribution on the instance space; then, with probability $c(x)$, the labeled example $(x, 1)$ is observed, and with probability $1 - c(x)$, the labeled example $(x, 0)$ is observed. Thus, in general, the learner has no direct access to the function c, even on individual points.

We view the learning problem as that of inferring from such randomly chosen examples a good approximation of the function c itself. Thus, we ask that the learner infer a real-valued hypothesis h for which $|h(x) - c(x)|$ is small for most instances x. This is called *learning with a model of probability.*

Specifically, we consider the problem of learning a class of real-valued read-once formulas, called *read-once real formulas.* Formulas in this class are constructed using two kinds of gates, or operators: The first gate, denoted MUL, simply computes the product of its real-valued inputs. The second gate, $\mathrm{LIN}_{z;w_1,\ldots,w_k}$, computes the weighted sum

$$\mathrm{LIN}_{z;w_1,\ldots,w_k}(y_1,\ldots,y_k) = z + w_1y_1 + \cdots + w_ky_k.$$

Here, k may be any positive integer, and z and the w_i's may be any real numbers which yield a function $\mathrm{LIN}_{z;w_1,\ldots,w_k}$ whose range is $[0,1]$ on the domain $[0,1]^k$.

Clearly, for a Boolean assignment to the input variables, a formula constructed from such gates outputs a real number between 0 and 1, and so these are indeed p-concepts. We show that this class can be learned with a model of probability against any product distribution (such as the uniform distribution).

Note that, for Boolean-valued inputs, the function MUL simply computes the logical AND of its inputs, and $\mathrm{LIN}_{1;-1}$ computes the logical negation of its single input. Thus, the class of read-once real formulas includes the class of read-once Boolean formulas with basis $\{\mathrm{AND},\mathrm{OR},\mathrm{NOT}\}$. Therefore, our result demonstrates for the first time the existence of a polynomial-time algorithm for inferring a good approximation of any such Boolean formula (against a product distribution).

Also, a unary gate $\mathrm{LIN}_{z;w}$ can alternatively be viewed as describing the behavior of a noisy or random Boolean gate which, on input 0 randomly outputs 1 with probability z, and on input 1 outputs 1 with probability $z + w$. (If the input to such a randomized gate is 1 with probability p, then the output is easily computed to be 1 with probability $\mathrm{LIN}_{z;w}(p) = z + wp$.) Thus, for the distributions considered, our result can be regarded as a demonstration of the learnability of read-once Boolean formulas, even when every gate and every wire of the formula is corrupted by significant amounts of randomness.

A similar probabilistic interpretation can be given to a non-unary gate $\mathrm{LIN}_{z;w_1,\ldots,w_k}$. Specifically, we can view such a gate as describing the probabilistic behavior of a kind of random "multiplexor" gate that, on inputs $y_1,\ldots,y_k$, does the following:

- with probability w_i, the gate outputs y_i;
- with probability z, the gate ignores its inputs and simply outputs 1;
- and with probability $1 - z - \sum w_i$, the gate ignores its inputs and outputs 0.

Thus, if input y_i is 1 with probability p_i, then the probability that such a randomized gate outputs 1 is $z + \sum w_ip_i$.

For instance, the binary gate $\mathrm{LIN}_{0;.5,.5}$ describes the behavior of a randomized gate that flips a fair coin and, if "heads" evaluates (and outputs) the left input, and if "tails" evaluates the right input.

Our result can therefore be viewed as a proof of the learnability of Boolean formulas constructed from the usual Boolean gates, in addition to the stochastic gates just described.

2. Preliminaries

In this paper, we will be exclusively interested in the learning of read-once formulas f constructed from *gates* (chosen from some basis), *variables* (denoted $x_1, \ldots, x_n$, and assumed in this paper to be Boolean) and, possibly, constants. We use the term *node* to refer to either a gate or a variable of some formula f. Note that because we consider only read-once formulas, there is a unique path from any node of f to the output. We define the *level* or *depth* of a node λ to be the number of gates (not including λ itself) on the path from λ to the output. Thus, the output node is at level 0. The *depth* of the entire formula is the maximum level of any node.

We say that a node λ *feeds* a gate λ' if the path from λ to the output goes through λ'. If λ is an input to λ', then we say that λ *immediately feeds* λ'. For any two variables x_i and x_j we define Γ_{ij} to be the deepest gate λ fed by both x_i and x_j. We say that a pair of variables x_i and x_j *meet* at the gate Γ_{ij}. We also write g_{ij} to denote the subformula subsumed by gate Γ_{ij}. (A subformula g is *subsumed* by node λ if λ is the root or output node of g.)

For $q_j \in \{0,1\}$ and $i_j \in \{1, \ldots, n\}$, $1 \leq j \leq r$, we write $f|x_{i_1} \leftarrow q_1, \ldots, x_{i_r} \leftarrow q_r$ to denote the function obtained from f by fixing or hard-wiring each variable x_{i_j} to the value q_j.

In our framework, the learner is attempting to infer an unknown *target concept* c chosen from some known *concept class* $\mathcal{C}$. In this paper, $\mathcal{C} = \bigcup_{n \geq 1} \mathcal{C}_n$ is parameterized by the number of variables n, and each $c \in \mathcal{C}_n$ represents a Boolean function on the domain $\{0,1\}^n$.

In the *distribution-free* or *probably approximately correct* (PAC) learning model, introduced by Valiant (1984), the learner is given access to labeled (positive and negative) examples of the target concept, drawn randomly and independently according to some unknown *target distribution* D. The learner is also given as input positive real numbers ϵ and δ. The learner's goal is to output with probability at least $1 - \delta$ a *hypothesis* h that has probability at most ϵ of disagreeing with c on a randomly drawn example from D (thus, the hypothesis has *accuracy* at least $1 - \epsilon$, and is said to be *ϵ-good*). If such a learning algorithm exists (that is, a polynomial-time algorithm meeting the goal for any $n \geq 1$, any $c \in \mathcal{C}_n$, any distribution D, and any positive values of ϵ and δ), we say that $\mathcal{C}$ is *PAC-learnable.* In this setting, polynomial time means polynomial in n, $1/\epsilon$ and $1/\delta$. In this paper, we will be primarily interested in a variant of Valiant's model in which the target distribution is known a priori to belong to a specific restricted class of distributions.

As described in detail by Kearns and Schapire (1990), a *probabilistic concept (p-concept)* is a function $c : \{0,1\}^n \rightarrow [0,1]$. The interpretation here is that $c(x)$ is the probability that an instance $x \in \{0,1\}^n$ is labeled 1, and $1 - c(x)$ is the probability it is labeled 0. Thus, we assume an oracle *EX* that first chooses x randomly according to some target distribution D, and then randomly labels x according to c as just described.

In this paper, we will be interested in the problem of *learning with a model of probability.* Here, the goal is to infer a good approximation of the function c itself.

Specifically, given positive ϵ and δ, we ask that, with probability at least $1 - \delta$, the learning algorithm find an *ϵ-good model of probability* for f, i.e., a real-valued hypothesis h such that

$$\mathbf{E}_{x \in D}\left[|h(x) - c(x)|\right] \leq \epsilon. \tag{1}$$

Furthermore, the learning algorithm's running time must be polynomial in $1/\epsilon$, $1/\delta$ and n. (This definition differs slightly, but is equivalent to, the definition of learning with a model of probability given by Kearns and Schapire (1990).)

We describe an algorithm for learning with a model of probability any p-concept in a particular class of p-concepts against any *product distribution* on the domain $\{0,1\}^n$, i.e., any distribution in which the setting of each bit x_i is chosen independently of the settings of the other bits.

The p-concept class of interest is the class of real-valued read-once formulas over the basis $\{\mathrm{MUL}, \mathrm{LIN}_{z;w_1,\ldots,w_k}\}$ where MUL denotes ordinary multiplication of two or more real numbers, and $\mathrm{LIN}_{z;w_1,\ldots,w_k}$ is the k-ary operator

$$\mathrm{LIN}_{z;w_1,\ldots,w_k}(y_1,\ldots,y_k) = z + \sum_{i=1}^{k} w_i y_i.$$

Here, k may be any positive integer, and z and the w_i's may be any real numbers which yield a function $\mathrm{LIN}_{z;w_1,\ldots,w_k}$ having range $[0,1]$ on the domain $[0,1]^k$ (or, equivalently, on the domain $\{0,1\}^k$). We call formulas over this basis *real formulas*. For instance,

$$\begin{aligned}\mathrm{LIN}_{.4;.3,.2}(\mathrm{MUL}(\mathrm{LIN}_{.5;.1,-.4}(x_1,x_2), \mathrm{LIN}_{1;-1}(x_3)), x_4) &= \\ .4 + .3 \cdot (.5 + .1x_1 - .4x_2) \cdot (1 - x_3) + .2x_4 & \end{aligned} \tag{2}$$

is a read-once real formula. Such formulas may also contain constants in the range $[0,1]$.

An easy induction argument shows that real formulas have range $[0,1]$, and so are p-concepts. Also, note that for Boolean-valued inputs, MUL is equivalent to AND, and $\mathrm{LIN}_{1;-1}$ is equivalent to NOT. Thus, the class of read-once real formulas includes the class of read-once Boolean formulas over the basis $\{\mathrm{AND}, \mathrm{NOT}\}$, or, equivalently, $\{\mathrm{AND}, \mathrm{OR}, \mathrm{NOT}\}$.

Further, our criterion (1) for learning of real formulas implies the usual PAC learnability of Boolean formulas: if c is a deterministic concept (i.e., a p-concept with range in $\{0,1\}$), and h is a real-valued hypothesis satisfying (1), then $h' = \mathrm{round}(h) = \lfloor h + 1/2 \rfloor$ is a 2ϵ-good hypothesis for c since

$$\mathbf{Pr}_{x \in D}\left[h'(x) \neq c(x)\right] \leq \mathbf{Pr}_{x \in D}\left[|h(x) - c(x)| \geq 1/2\right]$$

and, by Markov's inequality,

$$\epsilon \geq \mathbf{E}_{x \in D}\left[|h(x) - c(x)|\right] \geq \tfrac{1}{2}\mathbf{Pr}_{x \in D}\left[|h(x) - c(x)| \geq 1/2\right].$$

Note that the class of read-once real formulas also includes the class of Boolean formulas which have been corrupted with a kind of random misclassification noise in which a "true" output of 0 or 1 is misclassified with probability η_0 or η_1, respectively. This is because such noise can be simulated by a single, output-level gate $\mathrm{LIN}_{\eta_0,1-\eta_0-\eta_1}$: on input 0, this gate outputs η_0, and on input 1, it outputs $1-\eta_1$. Thus, η_0 and η_1 are the respective probabilities that an input of 0 or 1 is "misclassified" or flipped by this gate. (This noise model is considered by Goldman, Kearns and Schapire (1990), and is a slight generalization of the noise model considered by Sloan (1988) and Angluin and Laird (1988).)

Thus, our result can be viewed as a demonstration of the learnability of read-once Boolean formulas with large doses of noise sprinkled throughout the formulas. Such noise may affect the formula's output ("misclassification noise"), the inputs ("attribute noise") or it might affect the output of every gate of the formula.

In general, we can regard a gate $\mathrm{LIN}_{z;w_1,\ldots,w_k}$ as describing the behavior of a "noisy" or randomized Boolean gate as described in Section 1. If the ith input is 1 with probability p_i, then the gate outputs 1 with probability

$$z+\sum_{i=1}^{k} w_i p_i.$$

Thus, the probabilistic behavior of a formula constructed with such randomized gates is described by the p-concept obtained by replacing each randomized gate by a LIN gate, for appropriate choices of the LIN-gate parameters. (This can be proved rigorously, for instance, using a straightforward induction argument on the depth of the formula.)

3. The learning algorithm

Our learning procedure operates in three stages. In Stage I, we estimate the covariance of the formula f with each variable x_i. This gives a very rough measure of x_i's correlation with f. Intuitively, we expect a highly correlated variable to give a lot of information about f, while a poorly correlated variable is likely to be essentially useless. This intuition turns out to be correct, and in Stage I, all variables with small covariances are identified, discarded and ignored in later stages. (Note, however, that the correctness of this intuition depends critically on the nature of both the formulas and the distributions we are considering.)

In Stage II, we use various statistical measures to construct an approximation of the target formula's topology or skeleton structure. By the *skeleton* of a real formula, we refer to the topological structure of the formula, including the type of each gate. In other words, the skeleton is the formula stripped of all its LIN-gate parameters. For instance, the formula in (2) has skeleton:

$$\mathrm{LIN}(\mathrm{MUL}(\mathrm{LIN}(x_1,x_2),\mathrm{LIN}(x_3)),x_4).$$

Finally, in Stage III, we approximate the LIN-gate parameters for the skeleton inferred in Stage II, again using assorted statistical measures.

3.1. Some preliminary facts

Let f be the target formula, and let D be the target distribution. In what follows, all expectations are taken with respect to distribution D. For a function g, we also sometimes write $\bar{g}$ to denote its expectation:

$$\bar{g} = \mathbf{E}[g] = \mathbf{E}_{x \in D}[g(x)].$$

We will be interested in the partial derivatives of f, which turn out to be central to our theory. We will be interested not only in partial derivatives with respect to variables x_i, but also with respect to subformulas g of f. The following lemma demonstrates some useful properties of $\partial f/\partial g$.

Lemma 1 *Let g be a non-constant subformula of f subsumed by some node λ. Then either $\partial f/\partial g$ or its negation $-\partial f/\partial g$ is a read-once real formula. Furthermore, no variable of g is relevant to $\partial f/\partial g$.*

Proof: Note that the lemma's statement is equivalent to the statements (1) that $\partial f/\partial g$ is either a nonnegative or a nonpositive function, and (2) that $|\partial f/\partial g|$ is a read-once real formula, none of whose variables are relevant to g.

We prove these statements by induction on the depth of λ in f. If the depth is zero, then $f = g$, and $\partial f/\partial g = 1$ so the lemma holds trivially.

Otherwise, suppose the output gate of f is a gate $\text{LIN}_{z;w_1,\ldots,w_k}$. Then

$$f = \text{LIN}_{z;w_1,\ldots,w_k}(f_1,\ldots,f_k) = z + w_1 f_1 + \cdots + w_k f_k,$$

for some subformulas $f_1,\ldots,f_k$. Since λ has positive depth, it must occur in exactly one of these subformulas, say f_1. Then, by the chain rule,

$$\frac{\partial f}{\partial g} = \frac{\partial f}{\partial f_1} \cdot \frac{\partial f_1}{\partial g} = w_1 \cdot \frac{\partial f_1}{\partial g}.$$

By inductive hypothesis, $\partial f_1/\partial g$ is a nonpositive or a nonnegative function; thus, $\partial f/\partial g$ is as well. Since $|\partial f_1/\partial g|$ is also assumed to be a read-once real formula, and since $|w_1| \leq 1$ (by the properties of LIN gates), it must be that

$$|\partial f/\partial g| = \text{LIN}_{0;|w_1|}(|\partial f_1/\partial g|)$$

is also a read-once real formula. Finally, if x_i is relevant to g, then x_i is not relevant to $\partial f/\partial g$ (since, by inductive hypothesis, it is not relevant to $\partial f_1/\partial g$). This proves the induction in this case.

Suppose now that the output gate of f is a MUL gate that computes the product of $k \geq 2$ subformulas $f_1,\ldots,f_k$. Again, assume without loss of generality that λ occurs in f_1. Applying the chain rule as before, we have that

$$\frac{\partial f}{\partial g} = \frac{\partial f}{\partial f_1} \cdot \frac{\partial f_1}{\partial g} = f_2 \cdots f_k \cdot \frac{\partial f_1}{\partial g}.$$

Since each of the subformulas f_i is a read-once real formula, and by our inductive hypothesis, it follows as before that $\partial f/\partial g$ is either a nonpositive or a nonnegative function, and that

$$\left|\frac{\partial f}{\partial g}\right| = \text{MUL}\left(f_2, \ldots, f_k, \left|\frac{\partial f_1}{\partial g}\right|\right)$$

is a read-once real formula. If x_i is relevant to g, then it does not occur in any of the subformulas $f_2, \ldots, f_k$, nor in $\partial f_1/\partial g$ (by inductive hypothesis); therefore, it is not relevant to $\partial f/\partial g$. Thus, the induction holds in this case as well. This proves the lemma. ■

In particular, this lemma shows that $|\partial f/\partial g|$ is always bounded by 1.

Note that the formula f could be "multiplied out" to give a polynomial over the variable set. This polynomial is linear in each variable x_i. This follows, for instance, from Lemma 1 which shows that x_i is not relevant to $\partial f/\partial x_i$, and, therefore, that $\partial^2 f/\partial x_i^2 = 0$. Thus, we can write

$$f = u + vx_i$$

for some functions u and v to which x_i is not relevant. We call u and v the *decomposition* of f in terms of x_i. In the same manner, f can be decomposed in terms of any subformula g, and so can be written $f = u + vg$ for some functions u and v that do not contain any variable relevant to g.

Clearly, if $f = u + vx_i$, then

$$\frac{\partial f}{\partial x_i} = v = (f|x_i \leftarrow 1) - (f|x_i \leftarrow 0). \tag{3}$$

Since, as proved above, $\partial f/\partial x_i$ is either nonpositive or nonnegative on all inputs,

$$|\mathbf{E}[\partial f/\partial x_i]| = \mathbf{E}[|\partial f/\partial x_i|] = \mathbf{E}[|(f|x_i \leftarrow 1) - (f|x_i \leftarrow 0)|].$$

This latter expression is a natural measure of the *influence* of x_i, the degree to which x_i's value affects the value of f.

Henceforth, we let $B_i = \partial f/\partial x_i$.

In addition to the partial derivatives of f, we will also be interested in various central moments of f. For instance, for each variable x_i, we will be interested in the *covariance* C_i of f and x_i:

$$C_i = \mathbf{E}\left[(f - \bar{f})(x_i - \bar{x_i})\right].$$

Speaking very loosely, C_i measures the degree to which f's behavior is correlated with x_i. We will later see that variables that are weakly correlated with f (as measured by the covariance) can be safely ignored by the learning algorithm without introducing significant error.

It is well known, and easily verified that

$$C_i = \mathbf{E}[fx_i] - \mathbf{E}[f] \cdot \mathbf{E}[x_i]. \tag{4}$$

Thus, if $f = u + vx_i$ as above, then

$$\begin{aligned} C_i &= \mathbf{E}[x_i \cdot (u + vx_i)] - \mathbf{E}[x_i] \cdot \mathbf{E}[u + vx_i] \\ &= \bar{x}_i \cdot \bar{u} + \bar{v} \cdot \mathbf{E}\left[x_i^2\right] - \bar{x}_i \cdot \bar{u} - \bar{v} \cdot \bar{x}_i^2 \\ &= \bar{v}\bar{x}_i(1 - \bar{x}_i) \\ &= \bar{B}_i\bar{x}_i(1 - \bar{x}_i) \end{aligned} \tag{5}$$

where we have used our assumption of independence among variables, and the fact that $x_i^2 = x_i$ since x_i is Boolean. Thus, C_i is "small" if x_i is "sticky" (i.e., is almost always 0 or 1), or if x_i has low influence.

Note that, by equation (4), C_i can be well approximated by empirically estimating $\bar{x}_i$, $\bar{f}$ and $\mathbf{E}[fx_i]$. For instance, $\bar{x}_i$ is just the probability that $x_i = 1$; similarly, $\mathbf{E}[fx_i]$ is the probability that an example is chosen that is labeled 1, and for which $x_i = 1$.

In Stages II and III, we will also be interested in the *trivariance* T_{ij} of f and distinct variables x_i and x_j. This is defined to be

$$T_{ij} = \mathbf{E}\left[(f - \bar{f})(x_i - \bar{x}_i)(x_j - \bar{x}_j)\right].$$

We will see that this value is quite important to our algorithm. For example, the type (MUL or LIN) of gate Γ_{ij} can be determined from T_{ij}.

By multiplying out the product inside the expectations and using the independence of x_i and x_j, it is straightforward to verify that

$$T_{ij} = \mathbf{E}[fx_ix_j] - \mathbf{E}[fx_i] \cdot \mathbf{E}[x_j] - \mathbf{E}[fx_j] \cdot \mathbf{E}[x_i] + \mathbf{E}[f] \cdot \mathbf{E}[x_i] \cdot \mathbf{E}[x_j]. \tag{6}$$

Thus, as was so for C_i, T_{ij} can be well approximated from empirical estimates of $\bar{x}_i$, $\bar{x}_j$, $\bar{f}$, $\mathbf{E}[fx_i]$, $\mathbf{E}[fx_j]$ and $\mathbf{E}[fx_ix_j]$.

Finally, just as C_i turned out to be related to the first partial derivative of f, so T_{ij} is related to the second partial derivative. Since f is linear in every variable, f can be written as

$$f = u_0 + u_1x_i + u_2x_j + u_3x_ix_j \tag{7}$$

for some functions u_0, u_1, u_2 and u_3 to which x_i and x_j are irrelevant. Clearly

$$u_3 = \frac{\partial^2 f}{\partial x_i \partial x_j}, \tag{8}$$

which we henceforth denote A_{ij}.

By an argument similar to that given above, it can be verified using equations (6), (7) and (8) that

$$T_{ij} = \bar{A}_{ij}\bar{x}_i(1-\bar{x}_i)\bar{x}_j(1-\bar{x}_j). \tag{9}$$

This relationship will be very useful in what follows.

In Lemma 1, we proved that either $\partial f/\partial x_i$ or its negation is a read-once real formula. Thus, in either case, the lemma implies that the second partial derivative $\partial^2 f/\partial x_i \partial x_j$ has the same property — either it, or its negation, is a read-once real formula.

3.2. Stage I: eliminating uncorrelated variables

As described above, in Stage I, uncorrelated variables are eliminated. Our algorithm begins by estimating the expected values of f, x_i, fx_i and fx_ix_j for all variables x_i and x_j. Applying Chernoff (1963) bounds, we see that a polynomial-size sample suffices to obtain estimates that, with probability at least $1-\delta$, are each within

$$\tau = \frac{\epsilon_0 \epsilon^3}{(20n)^5}$$

of the true expectation, where $\epsilon_0 = \min\{\epsilon, 1/n\}$. That is, for each of the random variables R listed above, we require that

$$\left|\mathbf{E}[R] - \hat{\mathbf{E}}[R]\right| \le \tau,$$

where $\hat{\mathbf{E}}[R]$ denotes the empirical mean of R derived from the random sample. Henceforth, we assume that all of the random variables given above have the desired accuracy. As just mentioned, this will be the case with probability at least $1-\delta$.

In turn, we use these estimates in Stage I to obtain estimates $\hat{C}_i$ of the covariance C_i of f and x_i. In particular, we let

$$\hat{C}_i = \hat{\mathbf{E}}[fx_i] - \hat{\mathbf{E}}[f] \cdot \hat{\mathbf{E}}[x_i].$$

Then we have

$$\begin{aligned}
&\left|C_i - \hat{C}_i\right| \qquad\qquad (10)\\
&= \left|\mathbf{E}[fx_i] - \mathbf{E}[f]\cdot\mathbf{E}[x_i] - \hat{\mathbf{E}}[fx_i] + \hat{\mathbf{E}}[f]\cdot\hat{\mathbf{E}}[x_i]\right|\\
&\le \left|\mathbf{E}[fx_i] - \hat{\mathbf{E}}[fx_i]\right| + \hat{\mathbf{E}}[f]\cdot\left|\mathbf{E}[x_i] - \hat{\mathbf{E}}[x_i]\right| + \mathbf{E}[x_i]\cdot\left|\mathbf{E}[f] - \hat{\mathbf{E}}[f]\right|\\
&\le 3\tau.
\end{aligned}$$

Let $\beta = \epsilon/5n$. If $\left|\hat{C}_i\right| \ge \beta + 3\tau$, we say that x_i is *correlated*; in this case, $|C_i| \ge \beta$. In later stages, uncorrelated variables are ignored; for these variables, $|C_i| \le \beta + 6\tau \le \epsilon/4n$.

We show next that uncorrelated variables can be ignored without introducing significant error. Let f' be the read-once real formula obtained from f by replacing some variable x_i by the constant $p_i = \bar{x}_i$. Then formula f' is just the p-concept obtained by regarding x_i as a hidden variable — if we ignore x_i's value, then f' describes the probability that an assignment to the other variables is labeled 1. Let u, v be the decomposition of f in terms of x_i so that $f = u + vx_i$. Then $f' = u + vp_i$, and so

$$f - f' = v(x_i - p_i) = (\partial f/\partial x_i)(x_i - p_i).$$

Thus, by independence and equation (5),

$$\mathbf{E}[|f - f'|] = \mathbf{E}[|\partial f/\partial x_i|] \cdot \mathbf{E}[|x_i - p_i|] = |\bar{B}_i| \cdot 2p_i(1 - p_i) = 2|C_i|.$$

Note that if x_i is uncorrelated, then this latter expression is at most $\epsilon/2n$.

Suppose $x_1, \ldots, x_s$ are the uncorrelated variables of f. Let $f_0 = f$, let f_i be obtained from f_{i-1} by replacing x_i with the constant p_i for $1 \leq i \leq s$, and let $f_{\mathrm{I}} = f_s$. Then

$$\mathbf{E}[|f - f_{\mathrm{I}}|] \leq \sum_{i=1}^{s} \mathbf{E}[|f_i - f_{i-1}|] \leq s\epsilon/2n \leq \epsilon/2,$$

since, by the preceding argument, $\mathbf{E}[|f_i - f_{i-1}|] \leq \epsilon/2n$.

Henceforth, we regard f_{I} as the target formula. Sampling according to f_{I} is achieved by simply ignoring the variables eliminated from f. In later stages, it is shown how to find a hypothesis $\hat{f}$ such that $\mathbf{E}\left[\left|\hat{f} - f_{\mathrm{I}}\right|\right] \leq \epsilon/2$, and thus $\mathbf{E}\left[\left|\hat{f} - f\right|\right] \leq \epsilon$.

Note that we can easily handle at this point the special case that *all* the variables of f are eliminated. For this case, f_{I} simply computes some constant function p. Since $\bar{f} = \bar{f}_{\mathrm{I}} = p$, we simply let $\hat{f} = \hat{\mathbf{E}}[f]$, our previously obtained estimate of $\bar{f}$. Then $\mathbf{E}\left[\left|\hat{f} - f_{\mathrm{I}}\right|\right] = \left|\hat{\mathbf{E}}[f] - \mathbf{E}[f]\right| \leq \tau \leq \epsilon/2$, and thus $\mathbf{E}\left[\left|f - \hat{f}\right|\right] \leq \epsilon$ as desired.

3.3. Stage II: inferring the formula's skeleton

Based on the results of Stage I, we can assume henceforth that none of the variables of f are uncorrelated. Thus, in this and all subsequent sections, f actually refers to the formula f_{I} of the previous section, and all of the variables discussed are assumed to be correlated.

We show in this section how a good approximation of the skeletal structure of f can be obtained. Our approximation σ' will be sufficiently similar to the true skeleton σ that the differences between the two structures will be provably inconsequential.

Note that the functional composition of two LIN gates is a LIN gate, and that $\mathrm{LIN}_{0;1}$ is the identity function. Note also that any constant-valued subformulas occurring in f can be eliminated by "absorbing" them into the rest of the formula, and that a single k-ary MUL gate can be replaced by a tree of binary MUL gates. These observations allow us to assume without loss of generality that no subformula of f (including f itself) is equal to a constant, and that the gates of f occur in alternating layers of LIN and binary MUL gates, the output gate being a LIN gate, and every variable an input to a LIN gate. Thus, the topology or skeleton of f is entirely determined first by the type of each gate, and second by the tree structure of f with respect to its non-unary gates. Therefore, in reconstructing f's skeleton, we will be interested first in determining the type of each gate, and second in determining which gates are fed by which other gates, and in particular, which of two non-unary gates occurs deeper in the formula.

Our algorithm uses two tests for determining which of two gates is deeper. After describing the two tests, as well as a method for identifying the type of each gate, we show how the gathered information can be combined to construct an approximate skeleton.

For any three variables x_i, x_j and x_k, we must have either that $\Gamma_{ij} = \Gamma_{ik} = \Gamma_{jk}$, or that exactly two of the gates Γ_{ij}, Γ_{ik} and Γ_{jk} are the same, and that the remaining gate is the deepest of the three. For instance, if Γ_{ij} is the deepest gate, then it must feed $\Gamma_{ik} = \Gamma_{jk}$. Our purpose then is to determine which of these gates is deepest. This will be determined using the statistical measures described above.

Recall that good estimates were made in Stage I of the C_i's using the empirical means of various random variables. In the same manner, we can use these empirical estimates and equation (6) to obtain an estimate $\hat{T}_{ij}$ of each T_{ij}. Specifically, we let

$$\hat{T}_{ij} = \hat{\mathbf{E}}[fx_ix_j] - \hat{\mathbf{E}}[fx_i] \cdot \hat{\mathbf{E}}[x_j] - \hat{\mathbf{E}}[fx_j] \cdot \hat{\mathbf{E}}[x_i] + \hat{\mathbf{E}}[f] \cdot \hat{\mathbf{E}}[x_i] \cdot \hat{\mathbf{E}}[x_j].$$

Then, by an argument similar to that used in proving the accuracy of $\hat{C}_i$, we have that

$$\left|\hat{T}_{ij} - T_{ij}\right| \leq 8\tau.$$

It will also be convenient to assume that each $\hat{C}_i$ and each $\hat{T}_{ij}$ is in the range $[-1, 1]$; since C_i and T_{ij} are known to be in this range, we make this assumption without loss of generality. ("Clamping" estimates in this range can only improve their accuracy.)

3.3.1. Determining gate types

Our first goal in constructing the skeleton of f is to determine the type of each gate. Specifically, for each pair of variables x_i and x_j, we would like to determine if Γ_{ij} is a LIN gate or a MUL gate. It turns out that this can be determined from our estimate of T_{ij}, as is shown in the following lemma.

Lemma 2 *Let x_i and x_j be distinct variables. Then the following hold:*

1. *If Γ_{ij} is a* LIN *gate, then $T_{ij} = 0$.*
2. *If Γ_{ij} is a* MUL *gate, then $|T_{ij}| \geq |C_i| \cdot |C_j|$.*

Proof: Suppose first that Γ_{ij} is a gate $\text{LIN}_{z;w_1,\ldots,w_k}$. Then we can write g_{ij} as the weighted sum of its inputs $g_1, \ldots, g_k$:

$$g_{ij} = z + w_1 g_1 + \cdots + w_k g_k.$$

Without loss of generality, assume that x_i and x_j are relevant to g_1 and g_2, respectively. Then,

$$\begin{aligned} A_{ij} &= \frac{\partial^2 f}{\partial x_i \partial x_j} \\ &= \frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial g_{ij}} \cdot \frac{\partial g_{ij}}{\partial g_2} \cdot \frac{\partial g_2}{\partial x_j}\right) \\ &= \frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial g_{ij}} \cdot w_2 \cdot \frac{\partial g_2}{\partial x_j}\right) = 0 \end{aligned}$$

since x_i is relevant to neither $\partial f / \partial g_{ij}$ nor g_2. Part 1 then follows from equation (9).

Suppose now that Γ_{ij} is a MUL gate. We write subformula g_{ij} as $g_1 g_2$, the product of its inputs. (Recall that Γ_{ij} is binary by assumption.) Suppose without loss of generality that x_i and x_j are relevant to g_1 and g_2, respectively. Then we have by the chain rule that

$$\begin{aligned} B_i &= \frac{\partial f}{\partial x_i} \\ &= \frac{\partial f}{\partial g_{ij}} \cdot \frac{\partial g_{ij}}{\partial g_1} \cdot \frac{\partial g_1}{\partial x_i} \\ &= \frac{\partial f}{\partial g_{ij}} \cdot \frac{\partial g_1}{\partial x_i} \cdot g_2, \end{aligned} \tag{11}$$

and similarly

$$B_j = \frac{\partial f}{\partial g_{ij}} \cdot \frac{\partial g_2}{\partial x_j} \cdot g_1. \tag{12}$$

Thus,

$$A_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial f}{\partial g_{ij}} \cdot \frac{\partial g_1}{\partial x_i} \cdot \frac{\partial g_2}{\partial x_j}. \tag{13}$$

Together, these imply that $|A_{ij}| \geq |B_i||B_j|$ since $|\partial f / \partial g_{ij}|$, g_1 and g_2 are real formulas with range $[0, 1]$. Applying equations (5) and (9), this proves part 2 of the lemma. ■

This lemma is easily applied to the problem of determining the type of Γ_{ij}: If Γ_{ij} is a LIN gate, then $T_{ij} = 0$ and so $\left|\hat{T}_{ij}\right|$ cannot exceed 8τ. Conversely, if Γ_{ij} is a MUL gate, then the lemma implies that

$$|T_{ij}| \geq |C_i||C_j| \geq \beta^2$$

since each variable is assumed to be correlated; thus, in this case, $\left|\hat{T}_{ij}\right| \geq \beta^2 - 8\tau > 8\tau$. Thus, the type of Γ_{ij} can be exactly determined from the magnitude of $\hat{T}_{ij}$ (given the assumption made above that all estimates have the desired accuracy). Our algorithm uses this technique to quickly ascertain the type of every non-unary gate in the formula.

3.3.2. Determining relative gate depth: the type test

The gate-type information collected in the preceding section is also helpful in determining the topology of the unknown formula. To be more precise, suppose that we would like to determine which of the gates Γ_{ij}, Γ_{ik} or Γ_{jk} occurs deepest in the formula. As noted above, at least two of these gates must be the same gate, and therefore must obviously be of the same type. The third gate may or may not be of the same type. If it is of a different type, then our algorithm will detect this difference and so will be able to conclude correctly that the gate of the odd type must in fact be the deepest of the three gates. For instance, if we find that Γ_{ik} and Γ_{jk} are MUL gates, but that Γ_{ij} is a LIN gate, then the only possibility is that $\Gamma_{ik} = \Gamma_{jk}$, and that Γ_{ij} is the deepest of the three gates. Of course, if all three gates have the same type, then the type test yields no information on the relative depth of the three gates.

Thus, after determining the type of each gate, our algorithm tests, for each ordered triple x_i, x_j and x_k, whether Γ_{ij} is of a different type from Γ_{ik} and Γ_{jk}. If it finds that this is the case, then the algorithm can correctly conclude that $\Gamma_{ik} = \Gamma_{jk}$, and therefore Γ_{ij} occurs deeper than this gate.

The results of all these type tests are organized in a directed graph G_0. The vertices of G_0 are unordered pairs $\{i, j\}$, for $i \neq j$. For each ordered triple of distinct indices i, j, k, our algorithm tests if Γ_{ij} has a different type from Γ_{ik} and Γ_{jk}. If this is the case, an edge is directed in G_0 from $\{i, j\}$ to $\{i, k\}$. Thus, as argued above, an edge is added to G_0 in this fashion only if Γ_{ij} is deeper in f than Γ_{ik}. Moreover, by transitivity, a path in G_0 from $\{i, j\}$ to $\{i', j'\}$ implies that Γ_{ij} is deeper than $\Gamma_{i'j'}$.

As noted above, the type test is one sided in the sense that if Γ_{ij}, Γ_{ik} and Γ_{jk} all have the same type, then nothing can be concluded about the relative depth of the three gates. However, the next lemma gives conditions under which the type test and its graph G_0 are guaranteed to give useful depth information.

Lemma 3 *Let x_i, x_j and x_k be distinct variables of f, and assume Γ_{ij} is deeper in f than $\Gamma_{ik} = \Gamma_{jk}$. Suppose that either of the following conditions holds:*

1. *Γ_{ij} and Γ_{ik} are of different types; or*
2. *Γ_{ij} and Γ_{ik} are of the same type, but there exists a non-unary gate $\Gamma_{i\ell}$ of a different type on the path in f from Γ_{ij} to Γ_{ik}.*

Then there exists a path in G_0 from $\{i,j\}$ to $\{i,k\}$.

Proof: That condition 1 implies an edge in G_0 from $\{i,j\}$ to $\{i,k\}$ follows immediately from the remarks made above. Applying this fact twice shows that condition 2 implies that $\langle\{i,j\},\{i,\ell\}\rangle$ and $\langle\{i,\ell\},\{i,k\}\rangle$ are both edges in G_0. ■

Thus, if Γ_{ij} is deeper than $\Gamma_{ik} = \Gamma_{jk}$, then the type test will succeed in making this determination unless every non-unary gate on the path in f from Γ_{ij} to Γ_{ik} is of the same type. If all of the gates on this path are LIN gates, then they can all be collapsed into a single LIN gate. In other words, since we assume that the gates of f occur in alternating levels of LIN/MUL gates, this "path" must actually consist of the single LIN gate $\Gamma_{ij} = \Gamma_{ik} = \Gamma_{jk}$.

If, on the other hand, all the non-unary gates on this path are MUL gates, then we cannot make such a collapse since there may be unary LIN gates along the path. Thus, the problem that remains is that of determining which of two MUL gates is deeper, specifically, in the case that the path from one gate to the other includes only MUL gates and unary LIN gates. This problem is (almost) solved in the next section using the estimated values $\hat{C}_i$ and $\hat{T}_{ij}$, which have already proved useful.

3.3.3. *Determining relative gate depth: the quotient test*

The second test used for determining the relative depth of two MUL gates is called the *quotient test*, and it has a flavor similar to that of the type test. In this section, and later in Stage III, we will be interested in the quantity

$$E_{ij} = \frac{C_i C_j}{T_{ij}}.$$

We will see that if Γ_{ij} occurs deeper than $\Gamma_{ik} = \Gamma_{jk}$ then $E_{ik} = E_{jk}$, but E_{ij} will differ significantly from E_{ik} in most cases (the exception will be dealt with below). Thus, the quantities E_{ij} will give a second method for determining which of the three gates Γ_{ij}, Γ_{ik}, Γ_{jk} is deepest in f.

The following lemma gives a useful characterization of the quantity E_{ij}.

Lemma 4 *Let x_i and x_j be distinct variables which meet at a* MUL *gate. Then*

$$E_{ij} = \mathbf{E}\left[\frac{\partial f}{\partial g_{ij}}\right] \cdot \bar{g}_{ij}.$$

Proof: In the proof of Lemma 2, expressions (11), (12) and (13) were derived for B_i, B_j and A_{ij}. From these, and equations (5) and (9), it follows that

$$E_{ij} = \frac{C_i C_j}{T_{ij}} = \frac{\bar{B}_i \bar{B}_j}{\bar{A}_{ij}} = \mathbf{E}\left[\frac{\partial f}{\partial g_{ij}}\right] \cdot \bar{g}_1 \bar{g}_2.$$

Here, as in Lemma 2, g_1 and g_2 are the inputs to Γ_{ij}. Since, by independence, $\bar{g}_{ij} = \bar{g}_1 \bar{g}_2$, this proves the lemma. ■

Thus, if $\Gamma_{ik} = \Gamma_{jk}$, then Lemma 4 shows that $E_{ik} = E_{jk}$.

To estimate the quantities E_{ij}, we naturally use

$$\hat{E}_{ij} = \frac{\hat{C}_i \hat{C}_j}{\hat{T}_{ij}}.$$

To show that this is a good estimate, we will need the following lemma on estimating quotients.

Lemma 5 *For b and $\hat{b}$ nonzero,*

$$\left|\frac{\hat{a}}{\hat{b}} - \frac{a}{b}\right| \leq \frac{|\hat{a} - a|}{\left|\hat{b}\right|} + \left|\frac{a}{b}\right| \cdot \frac{\left|\hat{b} - b\right|}{\left|\hat{b}\right|}.$$

Proof:

$$\left|\frac{\hat{a}}{\hat{b}} - \frac{a}{b}\right| = \left|\frac{\hat{a}}{\hat{b}} - \frac{a}{\hat{b}} + \frac{a}{\hat{b}} - \frac{a}{b}\right| \leq \frac{|\hat{a} - a|}{\left|\hat{b}\right|} + \left|\frac{ab - a\hat{b}}{b\hat{b}}\right| = \frac{|\hat{a} - a|}{\left|\hat{b}\right|} + \left|\frac{a}{b}\right| \cdot \frac{\left|\hat{b} - b\right|}{\left|\hat{b}\right|}. \blacksquare$$

Let $\gamma = 28\tau/\beta^2$.

Lemma 6 *For any two distinct variables x_i and x_j,* $\left|E_{ij} - \hat{E}_{ij}\right| \leq \gamma$.

Proof: Applying Lemma 5, we have that

$$\left|E_{ij} - \hat{E}_{ij}\right| \leq \frac{\left|C_i C_j - \hat{C}_i \hat{C}_j\right|}{\left|\hat{T}_{ij}\right|} + |E_{ij}| \cdot \frac{\left|T_{ij} - \hat{T}_{ij}\right|}{\left|\hat{T}_{ij}\right|}.$$

Since $|E_{ij}| \leq 1$ (by Lemmas 1 and 4), and since $|C_i|$ and $\left|\hat{C}_j\right|$ are also bounded by 1, this is at most

$$\frac{\left|C_i - \hat{C}_i\right| + \left|C_j - \hat{C}_j\right| + \left|T_{ij} - \hat{T}_{ij}\right|}{\left|\hat{T}_{ij}\right|} \leq \frac{14\tau}{\beta^2 - 8\tau} \leq \gamma$$

as claimed. Here, we have used Lemma 2 to lower bound $\left|\hat{T}_{ij}\right|$. ■

Thus, $\hat{E}_{ij}$ is a good estimate of E_{ij}, and in particular, if $\Gamma_{ik} = \Gamma_{jk}$ then $E_{ik} = E_{jk}$ and $\left|\hat{E}_{ik} - \hat{E}_{jk}\right| \leq 2\gamma$. Therefore, for every ordered triple of variables x_i, x_j and x_k, our algorithm tests if all of the following hold:

$$\left|\hat{E}_{ik} - \hat{E}_{jk}\right| \leq 2\gamma,$$

$$\left|\hat{E}_{ij} - \hat{E}_{ik}\right| > 2\gamma,$$

$$\left|\hat{E}_{ij} - \hat{E}_{jk}\right| > 2\gamma.$$

If all of these hold, then our algorithm can correctly conclude that $E_{ij} \neq E_{ik} = E_{jk}$ (since at least two of these values must be equal) and therefore that Γ_{ij} is deeper than $\Gamma_{ik} = \Gamma_{jk}$.

Such information is added to the graph G_0 used for the type test. Specifically, an edge is added from $\{i, j\}$ to $\{i, k\}$ whenever the quotient test implies that Γ_{ij} is deeper than Γ_{ik}.

As was the case for the type test, the quotient test is one sided: it may be that two or three of the differences $\left|\hat{E}_{ik} - \hat{E}_{jk}\right|$, $\left|\hat{E}_{ij} - \hat{E}_{ik}\right|$ and $\left|\hat{E}_{ij} - \hat{E}_{jk}\right|$ do not exceed 2γ. In this case, the quotient test fails to give any useful information about the relative depths of Γ_{ij}, Γ_{ik} and Γ_{jk}. However, this can only happen if $\hat{E}_{ij}$, $\hat{E}_{ik}$ and $\hat{E}_{jk}$ are so close that the difference between any two of these estimates is bounded by 4γ. This fact will help us later in handling the situation that neither the type nor the quotient test succeeds in discerning the relative depth of two gates.

3.3.4. Constructing the skeleton from G_0

Finally, we are ready to show how an "approximate" skeleton of f can be computed from the graph G_0. First, to reiterate what was pointed out above, an edge in G_0 from $\{i, j\}$ to $\{i, k\}$ indicates that Γ_{ij} *must* be deeper than Γ_{ik} (given our assumption that all estimates are good); however, if Γ_{ij} is deeper than Γ_{ik}, there need not be a corresponding edge in G_0 (i.e., both tests above may have failed).

We can construct another graph G_1 from G_0 with a slightly different property: Graph G_1 has the same vertex set as G_0, and is obtained from G_0 by adding all edges from $\{i, j\}$ to $\{i, k\}$, $j \neq k$ which do not "contradict" G_0, i.e., for which there does not exist a path in G_0 from $\{i, k\}$ to $\{i, j\}$.

Thus, if Γ_{ij} is deeper than Γ_{ik}, or if $\Gamma_{ij} = \Gamma_{ik}$ then G_1 must contain an edge from $\{i, j\}$ to $\{i, k\}$ since in this case, G_0 certainly will not contain a path in the opposite direction. Furthermore, if there exists a path in G_1 from $\{i, j\}$ to $\{k, \ell\}$, but no path in the reverse direction, then we can conclude that Γ_{ij} is deeper in f than $\Gamma_{k\ell}$. When there exist paths connecting these vertices in both directions — that is, when $\{i, j\}$ and $\{k, \ell\}$ are in the same strongly connected component of G_1 —

there is no information about which gate is deeper. In this case, we collapse all of the gates corresponding to vertices of this strongly connected component (including Γ_{ij} and $\Gamma_{k\ell}$). In the remainder of this section, we give an algorithm for forming a skeleton for f by collapsing gates in this manner and we show that the skeleton we obtain is a good approximation of the true skeleton.

We say two vertices of G_1 are *equivalent* (with respect to G_1) if they are in the same strongly connected component. There are two ways in which vertices $\{i,j\}$ and $\{k,\ell\}$ may be equivalent. First, they may be equivalent if $\Gamma_{ij} = \Gamma_{k\ell}$, in which case Γ_{ij} and $\Gamma_{k\ell}$ can and certainly should be collapsed into the same gate. Alternatively, as suggested above, $\{i,j\}$ and $\{k,\ell\}$ may be equivalent due to insufficient information regarding the relative depth of Γ_{ij} and $\Gamma_{k\ell}$. Note that by the results of Section 3.3.2, this can only be a problem for MUL gates which are separated by unary LIN gates.

We say that a LIN gate of f is *trapped* if it is immediately fed by some MUL gate Γ_{ij}, and it immediately feeds another MUL gate $\Gamma_{k\ell}$ for some indices i, j, k, ℓ for which $\{i,j\}$ and $\{k,\ell\}$ are equivalent. In other words, the LIN gate is trapped if it is "surrounded" above and below by gates whose indices are in the same strongly connected component of G_1. Trapped LIN gates represent holes in our algorithm's knowledge of the formula's structure. Fortunately, it turns out that these gates have so little effect on the formula's behavior that they can be safely ignored. Ignoring trapped LIN gates essentially is equivalent to collapsing the surrounding MUL gates into a single (multi-input) MUL gate.

Thus, as will be shown below, the strongly connected components of G_1 generally correspond to connected regions of f (where we view f as a graph whose vertices are the gates, and whose edges are the "wires" connecting the gates), and that these connected regions can be safely approximated by simple LIN or MUL gates.

Let σ denote the true skeleton of f. Recall that we have assumed without loss of generality that the MUL gates of σ are all binary, and that the gates of f occur in alternating layers of LIN/MUL gates, the output gate being a LIN gate, and every input variable immediately feeding a LIN gate.

Let σ' be another skeleton derived from σ by first deleting all trapped LIN gates, and by then collapsing now adjacent MUL gates. That is, the deletion of a trapped LIN gate necessarily results in one MUL gate immediately feeding another; in constructing σ', we collapse these two gates into a single, multi-input MUL gate, purely for the sake of convenience and uniformity.

Clearly, as was so for σ, the gates of σ' occur in alternating layers of MUL and LIN gates with LIN gates occurring at the output and input levels. However, unlike σ, MUL gates in σ' may have more than two inputs.

We let Γ'_{ij} denote that gate in σ' at which variables x_i and x_j meet. To formalize the transformation of σ into σ', we let $\rho(\Gamma)$ denote that node Γ' of σ' into which node Γ of σ is transformed. Specifically, it can be verified that ρ has the following properties:

- For non-unary gates Γ_{ij} of σ, $\rho(\Gamma_{ij}) = \Gamma'_{ij}$.

- All variables are mapped to themselves: $\rho(x_i) = x_i$.
- If Γ is a non-trapped unary LIN gate, then $\rho(\Gamma)$ is that gate immediately fed by $\rho(\lambda)$, where λ immediately feeds Γ.
- Finally, if Γ is a trapped LIN gate, then Γ is actually eliminated in the construction of σ'; nevertheless, we define $\rho(\Gamma)$ to be $\rho(\lambda)$ where λ is the MUL gate that immediately feeds Γ (or, equivalently, we could have used the MUL gate immediately fed by Γ).

Clearly, the function ρ maps onto the set of all nodes of σ'. Furthermore, ρ is one-to-one if its domain is restricted to non-trapped LIN gates and variables.

Let

$$\rho^{-1}(\Gamma') = \{\Gamma : \rho(\Gamma) = \Gamma'\}$$

be the set of σ-nodes mapped by ρ to Γ'. Note that, by construction of σ', every set $\rho^{-1}(\Gamma')$ is connected; that is, if λ_1 and λ_2 are both contained in this set, then all of the nodes along the unique path in f from λ_1 to λ_2 are also contained in the set. Thus, every set $\rho^{-1}(\Gamma')$ has a unique member of minimal depth. We call this minimal-depth node $\alpha(\Gamma')$, and we say that a σ-node Γ is an *apex node* if it is in the image of α.

We show next how the skeleton σ' can be constructed from G_1. Later, in Stage III, we will show that σ' can be used to construct a good approximation of f.

Figure 1 shows our algorithm for constructing σ' from G_1. The algorithm maintains a family of skeletons F. On each iteration of the loop, several of these skeletons are combined into one. Taking strongly connected components with no incoming edges at line 4 guarantees that the algorithm will start from the inputs and work bottom-up. We will show that only one skeleton remains in F upon termination.

The operator COND-LIN(s) used at lines 10 and 14 returns s if the output gate of s is a LIN gate, and LIN(s) otherwise. (In other words, a LIN gate is added to s, but only if necessary.) Also, the set rel(s) for skeleton s is the set of variables relevant to s. Finally, $G - H$ at line 12 denotes the graph obtained from G by deleting all vertices in H, and any edges incident to these deleted vertices.

We say a condition holds *at all times* if it holds between each iteration of the main loop.

The algorithm clearly halts since some vertex of G is removed on each iteration.

Lemma 7 *At all times, if s_1 and s_2 are distinct members of F, then* $\mathrm{rel}(s_1) \cap \mathrm{rel}(s_2) = \emptyset$. *Also,* $\bigcup_{s \in F} \mathrm{rel}(s) = \{x_1, \ldots, x_n\}$ *(assuming all of the variables are correlated).*

Proof: Let s_0 and E be as in the algorithm. Then, using the fact that $\mathrm{rel}(s_0) = \bigcup_{s \in E} \mathrm{rel}(s)$, the lemma follows by an easy induction on the number of iterations of the main loop. ■

```
Input:  the graph $G_1$, and the type of each gate of $f$
Output: a skeleton $\sigma'$ that approximates the true skeleton $\sigma$ of $f$
Procedure:
1  $G \leftarrow G_1$
2  $F \leftarrow \{x_i : 1 \leq i \leq n\}$
3  repeat while $G$ is not empty
4      find a strongly connected component $H$ of $G$ with no incoming edges
5      $E \leftarrow \{s \in F : \{i,j\} \in H$ and $x_i \in \mathrm{rel}(s)\}$
6      let $\{i,j\}$ be any node of $H$, and let $E = \{s_1, \ldots, s_{|E|}\}$
7      if $\Gamma_{ij}$ is a LIN gate then
8          $s_0 \leftarrow \mathrm{LIN}(s_1, \ldots, s_{|E|})$
9      else    $\{\Gamma_{ij}$ is a MUL gate$\}$
10         $s_0 \leftarrow \mathrm{MUL}(\text{COND-LIN}(s_1), \ldots, \text{COND-LIN}(s_{|E|}))$
11     $F \leftarrow (F - E) \cup \{s_0\}$
12     $G \leftarrow G - H$
13 end
14 output COND-LIN$(s)$, where $s$ is the only member of $F$
```

Figure 1. An algorithm for inferring a good skeleton from G_1.

Lemma 8 *At all times, if $\{i, j\}$ is not a vertex of G then $\{x_i, x_j\} \subset \mathrm{rel}(s)$ for some $s \in F$.*

Proof: Note that $\{i, j\}$ is removed from G at line 12 only if x_i and x_j are relevant to skeletons in E. ■

Lemma 9 *Upon termination of the main loop, $|F| = 1$.*

Proof: If, upon termination, F contained two skeletons s_1 and s_2, then each must have distinct relevant variables x_i and x_j, respectively, by Lemma 7. By Lemma 8, this implies $\{i, j\} \in G$, contradicting the fact that G is empty. ■

Lemma 10 *At all times, if $s \in F$ then s is a subskeleton of σ'.*

Proof: By induction on the number of iterations of the main loop. Initially, the lemma holds trivially.

Consider the state of the algorithm immediately before G is modified at line 12. Let s_0 and E be as in the algorithm. Let $\{k, \ell\} \in H$ be such that $\Gamma_{k\ell}$ is a gate of minimal depth in the set $\{\Gamma_{ij} : \{i, j\} \in H\}$. (That is, $\Gamma_{k\ell}$ does not feed any gate in this set.)

Claim 1: $\mathrm{rel}(s_0) = \mathrm{rel}(g_{k\ell})$.

Proof of Claim 1: Note first that, by definition of E,

$$\mathrm{rel}(s_0) = \{x_i : \{i, j\} \in H\}.$$

Thus, if $x_i \in \mathrm{rel}(s_0)$ then $\{i,j\} \in H$ for some j. Let T be the set

$$T = \{\{i',j'\} \in G : \Gamma_{i'j'} \text{ feeds } \Gamma_{k\ell} \text{ or } \Gamma_{i'j'} = \Gamma_{k\ell}\}.$$

We will show that $\{i,j\} \in T$; this implies that x_i is in the subformula subsumed by $\Gamma_{k\ell}$, and thus $x_i \in \mathrm{rel}(g_{k\ell})$.

If $\{i,j\} \notin T$, then since $\{k,\ell\} \in T$ and since $\{i,j\}$ and $\{k,\ell\}$ are equivalent, there must be a path in H from $\{i,j\}$ to $\{k,\ell\}$, and there must be an edge on this path from some node $\{i',j'\} \notin T$ to some other node $\{i',k'\} \in T$. This means that $\Gamma_{i'k'}$ feeds or is equal to $\Gamma_{k\ell}$, but $\Gamma_{i'j'}$ is not in $g_{k\ell}$, the subformula subsumed by $\Gamma_{k\ell}$. In particular, this means that $\Gamma_{i'k'}$ feeds $\Gamma_{i'j'}$. Moreover, $\Gamma_{k\ell}$, being the root of $g_{k\ell}$, must certainly be on the path from $\Gamma_{i'k'}$ to $\Gamma_{i'j'}$. Thus, $\Gamma_{i'k'}$ feeds (or is equal to) $\Gamma_{k\ell}$, which in turn feeds $\Gamma_{i'j'}$. Therefore, there exist paths in G_1 from $\{i',k'\}$ to $\{k,\ell\}$, and from $\{k,\ell\}$ to $\{i',j'\}$. Since an edge is directed from $\{i',j'\}$ to $\{i',k'\}$, this implies that $\{i',j'\}$ is equivalent to $\{k,\ell\}$, and so must be in H. However, this contradicts the definition of $\{k,\ell\}$ since $\{k,\ell\}$ is deeper than $\{i',j'\}$.

Thus, $\mathrm{rel}(s_0) \subset \mathrm{rel}(g_{k\ell})$.

Conversely, suppose $x_i \in \mathrm{rel}(g_{k\ell})$. Then $\Gamma_{i\ell}$ feeds or is equal to $\Gamma_{k\ell}$. Thus, there is a path in G_1 from $\{i,\ell\}$ to $\{k,\ell\}$. Since H is assumed to have no incoming edges, this implies that either $\{i,\ell\} \in H$ or $\{i,\ell\} \notin G$. If $\{i,\ell\} \notin G$, then by Lemma 8, $x_i \in \mathrm{rel}(s_0)$ since $x_\ell \in \mathrm{rel}(s_0)$. If $\{i,\ell\} \in H$, then $x_i \in \mathrm{rel}(s_0)$ by construction. Thus, $\mathrm{rel}(s_0) = \mathrm{rel}(g_{k\ell})$, proving Claim 1.

Let s' be the subskeleton of σ' subsumed by gate $\Gamma'_{k\ell} = \rho(\Gamma_{k\ell})$. We will show that $s_0 = s'$, completing the induction and proving the lemma.

Claim 2: $\mathrm{rel}(s_0) = \mathrm{rel}(s')$.

Proof of Claim 2: We prove this claim by showing that $\mathrm{rel}(s') = \mathrm{rel}(g_{k\ell})$. In fact, it suffices to show that $\Gamma_{k\ell}$ is an apex node, since $\mathrm{rel}(\lambda') = \mathrm{rel}(\alpha(\lambda'))$ for any σ'-node λ'.

First, if $\Gamma_{k\ell}$ is a LIN gate, then it must be an apex gate since it is the only member of $\rho^{-1}(\Gamma'_{k\ell})$.

Suppose then that $\Gamma_{k\ell}$ is a MUL gate, but not an apex node. Then $\Gamma_{k\ell}$ immediately feeds a trapped LIN gate, which in turn feeds some other MUL gate Γ_{ij} for which $\{i,j\}$ and $\{k,\ell\}$ are equivalent. But then $\{i,j\} \in H$, contradicting the definition of $\Gamma_{k\ell}$.

This proves Claim 2.

By inductive hypothesis, each skeleton in E is a subskeleton of σ'. Claim 2 implies that each of these skeletons is in fact a subskeleton of s'.

Suppose x_i and x_j are relevant variables, respectively, of two distinct skeletons in E. Then $\{i,j\} \in G$, by Lemmas 7 and 8. Also, Γ_{ij} must feed or equal $\Gamma_{k\ell}$ by Claim 1. Thus, there must be a path in G from $\{i,j\}$ to $\{k,\ell\}$, and therefore, $\{i,j\}$ must be in H.

We claim that Γ_{ij} and $\Gamma_{k\ell}$ must be of the same type. Suppose not. Then there must be a gate $\Gamma_{ij'}$ that immediately feeds a gate $\Gamma_{ik'}$ of a different type, and such that both gates are on the inclusive path from Γ_{ij} to $\Gamma_{k\ell}$. Since there is a path in H from $\{k,\ell\}$ to $\{i,j\}$, there must be some edge in H directed from $\{i',k''\}$ to

$\{i', j''\}$ where these indices are such that $\Gamma_{ij'}$ feeds $\Gamma_{i'k''}$, and $\Gamma_{i'j''}$ feeds or equals $\Gamma_{ij'}$. However, because there is a path in f from $\Gamma_{i'j''}$ to $\Gamma_{i'k''}$ that passes through $\Gamma_{ij'}$ and $\Gamma_{ik'}$, and because these four gates cannot all be of the same type, we can conclude using Lemma 3 that there must be a path in G_0 from $\{i', j''\}$ to $\{i', k''\}$. But this contradicts the existence of the edge in G_1 from $\{i', k''\}$ to $\{i', j''\}$.

Therefore, Γ_{ij} and $\Gamma_{k\ell}$ must be of the same type. This implies that Γ'_{ij} and $\Gamma'_{k\ell}$ are also of the same type as $\Gamma_{k\ell}$. If $\Gamma_{k\ell}$ is a LIN gate, we have shown then that every gate on the path from Γ_{ij} to $\Gamma_{k\ell}$ is a LIN gate, and thus $\Gamma_{ij} = \Gamma_{k\ell}$, which implies $\Gamma'_{ij} = \Gamma'_{k\ell}$.

If $\Gamma_{k\ell}$ is a MUL gate, we have shown that every non-unary gate on the path from Γ_{ij} to $\Gamma_{k\ell}$ is a MUL gate, and furthermore, that every LIN gate on this path is trapped (since $\{i, j\}$ and $\{k, \ell\}$ are equivalent). Therefore, in this case as well, $\Gamma'_{ij} = \Gamma'_{k\ell}$.

This proves that $s' = s_0$, as desired, and completes the induction. ■

Combining Lemmas 7, 9, and 10, it follows immediately that the algorithm of Figure 1 outputs σ'. The algorithm clearly runs in polynomial time.

3.4. Stage III: inferring the LIN*-gate parameters*

In the final stage, our algorithm "fills in" the missing LIN-gate parameters for the skeleton inferred in Stage II.

The first problem we face is that σ' is only an approximation of the true skeleton σ, to which we have no direct access. Nevertheless, we will be able to show that the procedure we describe for constructing a complete formula from a skeleton yields the same result whether applied to the true skeleton σ or the approximate skeleton σ'. This fact will allow us to assume in proving the correctness of the procedure that we are actually dealing with the true skeleton σ.

Our algorithm computes the LIN-gate parameters of f from the bottom up: the algorithm visits each node of the skeleton, starting with variables. No node is visited until all those that feed it have been visited. When some node λ is visited which subsumes subformula g, we might like to compute a formula that approximates the function g; unfortunately, this does not seem to be possible because a given subformula may not be uniquely defined: by rearranging the LIN-gate parameters which occur in f, we can change the identity of individual subformulas (by constant factors) without altering the functional behavior of f. For instance, although the formulas

$$\mathrm{LIN}_{.1;-.8}(\mathrm{MUL}(\mathrm{LIN}_{.2;.1,.2}(x_1, x_2), \mathrm{LIN}_{0;.3}(x_3))$$

and

$$\mathrm{LIN}_{.1;-.2}(\mathrm{MUL}(\mathrm{LIN}_{.4;.2,.4}(x_1, x_2), \mathrm{LIN}_{0;.6}(x_3))$$

are equal (viewed as functions), they clearly contain many pairs of corresponding subformulas which are not equal.

Instead then, our algorithm infers an approximation of a function closely related to g, namely,

$$h = \mathbf{E}\left[\partial f/\partial g\right] \cdot g.$$

We call h the *canonical form* of g. Unlike g, the canonical form h is uniquely determined. Note also that if g is the entire formula f, then $h = g = f$.

Another nice property of the canonical form is that its expected value $\mathbf{E}\left[\partial f/\partial g\right] \cdot \bar{g}$ can be statistically approximated (given f's skeleton), as will be seen below. We write $\chi(\lambda)$ to denote this expectation. The next lemma yields a technique for estimating $\chi(\lambda)$, a key quantity used by our reconstruction procedure.

Recall from Section 3.3.3 that E_{ij} is defined to be $C_i \cdot C_j / T_{ij}$.

Lemma 11 *Let Γ_{ij} be a* MUL *gate of f.*

1. $\chi(\Gamma_{ij}) = E_{ij}$.
2. *If λ immediately feeds Γ_{ij}, then $\chi(\lambda) = \chi(\Gamma_{ij})$.*
3. *For any variable x_i, $\chi(x_i) = C_i/(1 - \bar{x}_i)$.*

Proof: Part 1 is simply a restatement of Lemma 4.

Let g be the subformula subsumed by λ. Then we can write $g_{ij} = g \cdot g'$ for some subformula g'. Then by independence and the chain rule,

$$\begin{aligned} \chi(\lambda) &= \mathbf{E}\left[\partial f/\partial g\right] \cdot \bar{g} \\ &= \mathbf{E}\left[\partial f/\partial g_{ij}\right] \cdot \mathbf{E}\left[\partial g_{ij}/\partial g\right] \cdot \bar{g} \\ &= \mathbf{E}\left[\partial f/\partial g_{ij}\right] \cdot \bar{g}'\bar{g} = \chi(\Gamma_{ij}). \end{aligned}$$

Finally, by equation (5), $\chi(x_i) = \mathbf{E}\left[\partial f/\partial x_i\right] \cdot \bar{x}_i = C_i/(1 - \bar{x}_i)$. ■

We use Lemma 11 to guide us in assigning an estimate $\hat{\chi}(\lambda')$ to every node of σ'. Specifically, if λ' is a MUL gate, then we set $\hat{\chi}(\lambda') = \hat{E}_{k\ell}$ where k, ℓ is an arbitrarily chosen pair for which $\Gamma'_{k\ell} = \lambda'$ in σ'. If λ' is a LIN gate which immediately feeds a MUL gate λ'_0, then we set $\hat{\chi}(\lambda') = \hat{\chi}(\lambda'_0)$. If λ' is an output node, then we let $\hat{\chi}(\lambda') = \hat{\mathbf{E}}\left[f\right]$. Finally, if λ' is a variable x_i, then we let $\hat{\chi}(\lambda') = \hat{C}_i/(1 - \hat{\mathbf{E}}\left[x_i\right])$. (Recall that $\hat{\mathbf{E}}\left[x_i\right]$ and $\hat{\mathbf{E}}\left[f\right]$ were both estimated in computing $\hat{C}_i$ and $\hat{T}_{ij}$.)

The assignments made above are actually performed by our reconstruction algorithm. For the sake of argument, we also define $\hat{\chi}(\lambda)$, for every node λ of σ, to be $\hat{\chi}(\rho(\lambda))$.

We pause in our development to show that these estimates are fairly good. The following lemma will be helpful for this purpose.

Lemma 12 *Suppose* MUL-*gate* Γ_{ij} *immediately feeds a trapped* LIN *gate which in turn immediately feeds* MUL-*gate* Γ_{ik}. *Then* $|E_{ij} - E_{ik}| \leq 18\gamma$.

Proof: Since $\{i, j\}$ and $\{i, k\}$ are equivalent (by virtue of the trapped LIN gate), there must be some edge in G_1 directed from $\{i', k'\}$ to $\{i', j'\}$, and such that $\Gamma_{i'j'}$ feeds or is equal to Γ_{ij}, and Γ_{ik} feeds or is equal to $\Gamma_{i'k'}$. By possibly renaming variables, we can assume without loss of generality that $\Gamma_{ij} = \Gamma_{i'j}$ and $\Gamma_{ik} = \Gamma_{i'k}$.

There cannot be a path in G_0 from $\{i', j'\}$ to $\{i', k'\}$ since this would deny the possibility of an edge in G_1 in the reverse direction.

In particular, this implies that there is no edge in G_0 from $\{i', j'\}$ to $\{i', k'\}$, meaning that the quotient test failed for this triple. As pointed out in Section 3.3.3, such a failure implies that

$$\left|\hat{E}_{i'j'} - \hat{E}_{i'k'}\right| \leq 4\gamma.$$

Likewise, G_0 cannot include both of the edges $\langle\{i', j'\}, \{i', j\}\rangle$ and $\langle\{i', j\}, \{i', k'\}\rangle$, implying that either $\left|\hat{E}_{i'j'} - \hat{E}_{i'j}\right|$ or $\left|\hat{E}_{i'j} - \hat{E}_{i'k'}\right|$ is bounded by 4γ. Combined with the above argument, either case implies that

$$\left|\hat{E}_{i'j'} - \hat{E}_{i'j}\right| \leq 8\gamma.$$

A similar argument shows that

$$\left|\hat{E}_{i'j'} - \hat{E}_{i'k}\right| \leq 8\gamma.$$

Thus,

$$\left|\hat{E}_{i'j} - \hat{E}_{i'k}\right| \leq 16\gamma,$$

which implies by Lemma 6 that

$$|E_{ij} - E_{ik}| = |E_{i'j} - E_{i'k}| \leq 18\gamma. \qquad \blacksquare$$

Lemma 13 *Let* λ *be a node of* σ. *Then* $|\chi(\lambda) - \hat{\chi}(\lambda)| \leq 18\gamma n$.

Proof: There are four cases to consider:

Case 1 λ *is a variable* x_i.

In this case, by Lemmas 5 and 11,

$$\begin{aligned} |\chi(\lambda) - \hat{\chi}(\lambda)| &= \left|\frac{C_i}{1 - \mathbf{E}[x_i]} - \frac{\hat{C}_i}{1 - \hat{\mathbf{E}}[x_i]}\right| \\ &\leq \frac{\left|C_i - \hat{C}_i\right| + |\chi(\lambda)| \cdot \left|\mathbf{E}[x_i] - \hat{\mathbf{E}}[x_i]\right|}{1 - \hat{\mathbf{E}}[x_i]} \\ &\leq \frac{3\tau + \tau}{\beta - \tau} \leq 18\gamma n \end{aligned}$$

where we have made use of the assumed accuracy of $\hat{\mathbf{E}}[x_i]$ and $\hat{C}_i$, as well as the facts that $|\chi(\lambda)| \leq 1$, and that $1 - \hat{\mathbf{E}}[x_i] \geq 1 - \mathbf{E}[x_i] - \tau \geq \left|\hat{C}_i\right| - \tau \geq \beta - \tau$ (by equation (5)).

Case 2 *λ is some* MUL *gate Γ_{ij}.*

In this case, $\hat{\chi}(\lambda) = \hat{\chi}(\rho(\lambda)) = \hat{E}_{k\ell}$ for some pair k, ℓ for which $\Gamma'_{ij} = \Gamma'_{k\ell}$ (i.e., k, ℓ was the pair chosen arbitrarily by our algorithm).

Without loss of generality, let Γ_{ik} be the lowest common ancestor of Γ_{ij} and $\Gamma_{k\ell}$. Then $\Gamma'_{ik} = \Gamma'_{ij} = \Gamma'_{k\ell}$, and so every LIN gate is trapped on the paths from Γ_{ij} to Γ_{ik}, and from $\Gamma_{k\ell}$ to Γ_{ik}. Applying Lemma 12 repeatedly to each of the n_1 MUL gates on the (inclusive) path from Γ_{ij} to Γ_{ik}, we see that

$$|E_{ik} - E_{ij}| \leq 18\gamma(n_1 - 1).$$

Similarly,

$$|E_{ik} - E_{k\ell}| \leq 18\gamma(n_2 - 1),$$

where n_2 is the number of MUL gates on the path from $\Gamma_{k\ell}$ to Γ_{ik}. Thus,

$$\left|E_{ij} - \hat{E}_{k\ell}\right| \leq 18\gamma(n_1 + n_2 - 2) + \gamma.$$

Since $n_1 + n_2 - 1 \leq n$ (the maximum number of MUL gates in f), and since $\chi(\lambda) = E_{ij}$ and $\hat{\chi}(\lambda) = \hat{E}_{k\ell}$, the lemma follows in this case.

Case 3 *λ is a* LIN *gate which feeds some* MUL *gate λ_0.*

In this case, $\chi(\lambda) = \chi(\lambda_0)$, by Lemma 11. Also, by definition,

$$\hat{\chi}(\lambda) = \hat{\chi}(\rho(\lambda)) = \hat{\chi}(\rho(\lambda_0)) = \hat{\chi}(\lambda_0)$$

since $\rho(\lambda)$ immediately feeds $\rho(\lambda_0)$ in σ'. Thus,

$$|\chi(\lambda) - \hat{\chi}(\lambda)| = |\chi(\lambda_0) - \hat{\chi}(\lambda_0)|,$$

and the lemma follows in this case by the proof for Case 2.

Case 4 *λ is the output* LIN *gate.*

In this case, $\chi(\lambda) = \mathbf{E}[f]$, and $\hat{\chi}(\lambda) = \hat{\chi}(\rho(\lambda)) = \hat{\mathbf{E}}[f]$, since $\rho(\lambda)$ is the output gate of σ'. Thus,

$$|\chi(\lambda) - \hat{\chi}(\lambda)| = \left|\mathbf{E}[f] - \hat{\mathbf{E}}[f]\right| \leq \tau \leq 18\gamma n. \qquad \blacksquare$$

As mentioned, our reconstruction procedure works by visiting each node λ of the given skeleton from the bottom up, starting with the input variables, estimating at each node the canonical form h for the subformula g subsumed by λ. An estimate $\hat{h}$ of h is computed based on: (1) the node type of λ, (2) the function $\hat{\chi}$, and (3) the previously computed estimates $\hat{h}_i$ of the canonical forms of the subformulas which immediately feed λ.

Specifically, the algorithm works as follows, depending on the node type of λ.

Case 1 *λ is a variable x_i.*

In this case, $g = x_i$, and $h = \mathbf{E}[\partial f/\partial x_i] \cdot x_i = \bar{B}_i x_i$. Motivated by equation (5), we estimate $\bar{B}_i$ using

$$\hat{B}_i = \frac{\hat{C}_i}{\hat{\mathbf{E}}[x_i](1 - \hat{\mathbf{E}}[x_i])},$$

and we set $\hat{h} = \hat{B}_i x_i$.

Case 2 *λ is a* MUL *gate.*

In this case, λ computes the product of k subformulas whose canonical forms have previously been estimated by $\hat{h}_1, \ldots, \hat{h}_k$. (Although f contains only binary MUL gates, we will want to apply this procedure to σ' which may include non-unary MUL gates.) In this case, we set

$$\hat{h} = \frac{\hat{h}_1 \cdots \hat{h}_k}{(\hat{\chi}(\lambda))^{k-1}}.$$

(The motivation for this choice will be seen below.)

Case 3 *λ is a* LIN *gate.*

In this case, λ computes a weighted sum of k subformulas whose canonical forms were estimated by $\hat{h}_1, \ldots, \hat{h}_k$. Let $\lambda_1, \ldots, \lambda_k$ be the gates which immediately feed λ. Then we set

$$\hat{h} = \hat{\chi}(\lambda) + \sum_{i=1}^{k} (\hat{h}_i - \hat{\chi}(\lambda_i)).$$

(Again, motivation will be provided below.)

The first fact we will want to prove about this procedure is that it yields the same result whether applied to σ or σ'. This is proved in the lemmas below.

We say that a node λ in σ is an *apex-child* of another node λ_0 if λ is an apex node which feeds λ_0, and if there are no other apex nodes in σ on the path from λ to λ_0.

Also, we write $\hat{h}(\lambda)$ to denote the function $\hat{h}$ computed by the above procedure upon visiting gate λ. Here, it is understood that $\hat{h}(\lambda)$ was computed using the skeleton (σ or σ') to which λ belongs.

Lemma 14 *Let λ be a* MUL *gate or a trapped* LIN *gate of σ, and let $\lambda_1, \ldots, \lambda_k$ be its apex-children. Then*

$$\hat{h}(\lambda) = \frac{\hat{h}(\lambda_1) \cdots \hat{h}(\lambda_k)}{(\hat{\chi}(\lambda))^{k-1}}.$$

Proof: By induction on the maximum depth of the apex-children (relative to λ).

The lemma holds trivially in the degenerate base case that this depth is zero (i.e., $k = 1$).

Suppose first that λ is a trapped LIN gate immediately fed by MUL gate λ_0. By definition, $\rho(\lambda) = \rho(\lambda_0)$, so $\hat{\chi}(\lambda) = \hat{\chi}(\lambda_0)$. Thus, our procedure computes

$$\hat{h}(\lambda) = (\hat{h}(\lambda_0) - \hat{\chi}(\lambda_0)) + \hat{\chi}(\lambda) = \hat{h}(\lambda_0).$$

The lemma then follows in this case by our inductive hypothesis on λ_0, and because λ_0 has the same apex-children as λ.

Suppose now that λ is a MUL gate immediately fed by nodes Γ_1 and Γ_2. (Since λ is in σ, it must be binary.) Then without loss of generality, $\lambda_1, \ldots, \lambda_r$ are the apex-children of Γ_1, and $\lambda_{r+1}, \ldots, \lambda_k$ are the apex-children of Γ_2, for some $1 \leq r < k$. Then our procedure computes

$$\hat{h}(\lambda) = \frac{\hat{h}(\Gamma_1) \cdot \hat{h}(\Gamma_2)}{\hat{\chi}(\lambda)}$$

which, by inductive hypothesis, equals

$$\frac{\hat{h}(\lambda_1) \cdots \hat{h}(\lambda_k)}{\hat{\chi}(\lambda) \cdot (\hat{\chi}(\Gamma_1))^{r-1} \cdot (\hat{\chi}(\Gamma_2))^{k-r-1}}.$$

If $r > 1$, then Γ_1 cannot be an apex gate, which implies that $\rho(\Gamma_1) = \rho(\lambda)$ and so $\hat{\chi}(\Gamma_1) = \hat{\chi}(\lambda)$. Thus, for $r \geq 1$, $(\hat{\chi}(\Gamma_1))^{r-1} = (\hat{\chi}(\lambda))^{r-1}$. (Equality holds trivially when $r = 1$.)

Similarly, for $r \leq k - 1$, $(\hat{\chi}(\Gamma_2))^{k-r-1} = (\hat{\chi}(\lambda))^{k-r-1}$. Thus, as desired,

$$\hat{h}(\lambda) = \frac{\hat{h}(\lambda_1) \cdots \hat{h}(\lambda_k)}{(\hat{\chi}(\lambda))^{k-1}}. \qquad \blacksquare$$

Lemma 15 *Let λ be an apex node of σ. Then $\hat{h}(\lambda) = \hat{h}(\rho(\lambda))$.*

Proof: By induction on the depth of the subformula subsumed by λ. The result is trivial in the base case that λ is a variable.

Suppose λ is a LIN gate. Let $\lambda_1, \ldots, \lambda_k$ be the gates which immediately feed λ. Then each λ_i is an apex node, so by inductive hypothesis, $\hat{h}(\lambda_i) = \hat{h}(\rho(\lambda_i))$. Since the same computation is performed by the procedure in computing $\hat{h}(\lambda)$ and $\hat{h}(\rho(\lambda))$, the result follows trivially.

Finally, suppose λ is a MUL gate. Let $\lambda_1, \ldots, \lambda_k$ be the apex-children of λ. Then in σ', $\rho(\lambda_1), \ldots, \rho(\lambda_k)$ are exactly the gates that immediately feed $\rho(\lambda)$. Thus, by Lemma 14 and the inductive hypothesis, we have as desired

$$\hat{h}(\lambda) = \frac{\hat{h}(\lambda_1)\cdots\hat{h}(\lambda_k)}{(\hat{\chi}(\lambda))^{k-1}} = \frac{\hat{h}(\rho(\lambda_1))\cdots\hat{h}(\rho(\lambda_k))}{(\hat{\chi}(\rho(\lambda)))^{k-1}} = \hat{h}(\rho(\lambda)). \qquad \blacksquare$$

Thus, in particular, since the output gate of σ is an apex gate, Lemma 15 shows that the final formula computed by our procedure if applied to σ would be the same as that computed for σ'. Therefore, although the procedure would in fact be applied to σ', we will analyze its result as if it were computed using σ.

Suppose the procedure is visiting node λ, which subsumes subformula g with canonical form h. The procedure computes an estimate $\hat{h}$ of h. For analysis purposes, we define an estimator $\hat{g} = \hat{h}/\mathbf{E}[\partial f/\partial g]$. We will prove, by induction, that

$$\mathbf{E}[|g - \hat{g}|] \le \frac{(2s-1)\epsilon_0}{16n}$$

where s is the number of nodes subsumed by λ (inclusive). Since, when λ is the output node, $f = g = h$ and $\hat{g} = \hat{h}$, and since an easy induction argument shows that there are at most $4n - 2$ nodes in the entire formula f, this will prove that the final output formula $\hat{f}$ is such that

$$\mathbf{E}\left[\left|f - \hat{f}\right|\right] \le \frac{(8n-5)\epsilon_0}{16n} \le \frac{\epsilon}{2}.$$

As usual, there are several cases to consider.

Case 1 *λ is a variable x_i.*

In this case, $g = x_i$, and, as noted above, $h = \bar{B}_i x_i$, and $\hat{h} = \hat{B}_i x_i$. Then

$$\mathbf{E}[|g - \hat{g}|] = \mathbf{E}\left[\left|x_i - \frac{\hat{B}_i x_i}{\bar{B}_i}\right|\right] \le \left|1 - \frac{\hat{B}_i}{\bar{B}_i}\right| = \frac{\left|\bar{B}_i - \hat{B}_i\right|}{\left|\bar{B}_i\right|}. \qquad (14)$$

Let $q = \mathbf{E}[x_i](1 - \mathbf{E}[x_i])$ and let $\hat{q} = \hat{\mathbf{E}}[x_i](1 - \hat{\mathbf{E}}[x_i])$. Then

$$|q - \hat{q}| \le 2\left|\mathbf{E}[x_i] - \hat{\mathbf{E}}[x_i]\right| \le 2\tau.$$

Using Lemma 5, we have that (14) is at most

$$\begin{aligned} \frac{1}{\left|\bar{B}_i\right|} \cdot \left(\frac{\left|C_i - \hat{C}_i\right|}{\hat{q}} + \left|\bar{B}_i\right| \cdot \frac{|q - \hat{q}|}{\hat{q}}\right) &\le \frac{3\tau + 2\tau}{\left|\bar{B}_i\right| \cdot (q - 2\tau)} \\ &\le \frac{5\tau}{|C_i| - 2\tau} \le \frac{10\tau}{\beta} \le \frac{\epsilon_0}{16n} \end{aligned}$$

since $C_i = \bar{B}_i q$, and $2\tau \le \frac{1}{2}\beta \le \frac{1}{2}|C_i|$. Thus, the inductive hypothesis is satisfied in this case since there is only one node subsumed by λ.

Case 2 *λ is a* MUL *gate.*

Since λ is a MUL gate of σ, it must be binary, and so must be immediately fed by two nodes λ_1 and λ_2; these nodes subsume two subformulas g_1 and g_2 (respectively), so $g = g_1 g_2$. Let h_1 and h_2 be the canonical forms of g_1 and g_2. We have that

$$\begin{aligned} h_1 &= \mathbf{E}\left[\partial f/\partial g_1\right] \cdot g_1 \\ &= \mathbf{E}\left[\partial f/\partial g\right] \cdot \mathbf{E}\left[\partial g/\partial g_1\right] \cdot g_1 \\ &= \mathbf{E}\left[\partial f/\partial g\right] \cdot \bar{g}_2 g_1. \end{aligned}$$

Similarly,

$$h_2 = \mathbf{E}\left[\partial f/\partial g\right] \cdot \bar{g}_1 g_2.$$

Thus,

$$h = \mathbf{E}\left[\partial f/\partial g\right] \cdot g = \frac{h_1 h_2}{\mathbf{E}\left[\partial f/\partial g\right] \cdot \bar{g}} = \frac{h_1 h_2}{\chi(\lambda)}.$$

Note that, if $\hat{h}_1$ and $\hat{h}_2$ are the previously computed estimates of h_1 and h_2, then our procedure computes

$$\hat{h} = \frac{\hat{h}_1 \hat{h}_2}{\hat{\chi}(\lambda)}.$$

For ease of notation, let $K = \mathbf{E}\left[\partial f/\partial g\right]$, let $e = \chi(\lambda)$ and let $\hat{e} = \hat{\chi}(\lambda)$. We would like to compute the error of $\hat{g} = \hat{h}/K$. We assume inductively that the estimates $\hat{g}_1$ and $\hat{g}_2$ are such that

$$\mathbf{E}\left[|g_i - \hat{g}_i|\right] \le \frac{(2s_i - 1)\epsilon_0}{16n}$$

where s_i is the number of nodes in g_i. By definition, $\hat{h}_1 = K\bar{g}_2\hat{g}_1$ and $\hat{h}_2 = K\bar{g}_1\hat{g}_2$. Thus,

$$\begin{aligned} \hat{g} &= \frac{\hat{h}_1 \hat{h}_2}{K\hat{e}} \\ &= \frac{(K\bar{g}_2\hat{g}_1)(K\bar{g}_1\hat{g}_2)}{K\hat{e}} \\ &= \frac{K\bar{g}_1\bar{g}_2}{\hat{e}} \cdot \hat{g}_1\hat{g}_2 \\ &= \frac{e}{\hat{e}} \cdot \hat{g}_1\hat{g}_2. \end{aligned}$$

So we can analyze the error of $\hat{g}$ as:

$$\begin{aligned}\mathbf{E}\left[|g-\hat{g}|\right] &= \mathbf{E}\left[\left|\frac{e}{\hat{e}}\hat{g}_1\hat{g}_2 - g_1g_2\right|\right] \\ &\le \left|1-\frac{e}{\hat{e}}\right| \cdot \mathbf{E}\left[\hat{g}_1\hat{g}_2\right] + \mathbf{E}\left[|g_1g_2 - \hat{g}_1\hat{g}_2|\right]. \qquad (15)\end{aligned}$$

We have that $|e-\hat{e}| \le 18\gamma n \le \frac{1}{2}\beta^2 \le \frac{1}{2}|e|$ by Lemmas 11 and 13. Thus, the first term of (15) can be bounded by

$$\begin{aligned}\frac{|e-\hat{e}|}{|\hat{e}|} \cdot \mathbf{E}\left[\hat{g}_1\hat{g}_2\right] &\le \frac{|e-\hat{e}|}{|e|-|e-\hat{e}|} \cdot \mathbf{E}\left[\hat{g}_1\hat{g}_2\right] \\ &\le \frac{2|e-\hat{e}|}{|e|} \cdot \mathbf{E}\left[\hat{g}_1\hat{g}_2\right] \\ &\le \frac{2|e-\hat{e}|}{|K|\bar{g}_1\bar{g}_2} \cdot \left(\bar{g}_1\bar{g}_2 + \mathbf{E}\left[|g_1g_1 - \hat{g}_1\hat{g}_2|\right]\right). \qquad (16)\end{aligned}$$

Also,

$$\begin{aligned}&\mathbf{E}\left[|g_1g_2 - \hat{g}_1\hat{g}_2|\right] \le \\ &\qquad \bar{g}_1 \cdot \mathbf{E}\left[|g_2-\hat{g}_2|\right] + \bar{g}_2 \cdot \mathbf{E}\left[|g_1-\hat{g}_1|\right] + \mathbf{E}\left[|g_1-\hat{g}_1|\right] \cdot \mathbf{E}\left[|g_2-\hat{g}_2|\right].\end{aligned}$$

Thus, expression (16) is at most

$$2|e-\hat{e}| \cdot \left(\frac{1}{|K|} + \frac{\mathbf{E}\left[|g_1-\hat{g}_1|\right]}{|K|\bar{g}_1} + \frac{\mathbf{E}\left[|g_2-\hat{g}_2|\right]}{|K|\bar{g}_2} + \frac{\mathbf{E}\left[|g_1-\hat{g}_1|\right] \cdot \mathbf{E}\left[|g_2-\hat{g}_2|\right]}{|K|\bar{g}_1\bar{g}_2}\right).$$

Note that

$$|K| = |\mathbf{E}\left[\partial f/\partial g\right]| \ge |\mathbf{E}\left[\partial f/\partial x_i\right]| \ge |C_i| \ge \beta,$$

where x_i is any variable relevant to g. Likewise, $|K\bar{g}_1| = |\mathbf{E}\left[\partial f/\partial g_2\right]| \ge \beta$, and similarly, $|K\bar{g}_2| \ge \beta$.

Combining facts and rearranging terms, this shows that expression (15) is at most

$$\begin{aligned}&\frac{36\gamma n}{\beta} + \left(1+\frac{36\gamma n}{\beta}\right) \cdot \left(\mathbf{E}\left[|g_1-\hat{g}_1|\right] + \mathbf{E}\left[|g_2-\hat{g}_2|\right]\right) \\ &\qquad + \left(1+\frac{36\gamma n}{\beta^2}\right) \cdot \mathbf{E}\left[|g_1-\hat{g}_1|\right]\mathbf{E}\left[|g_2-\hat{g}_2|\right] \\ &\le \frac{36\gamma n}{\beta} + \left(1+\frac{36\gamma n}{\beta}\right) \cdot \frac{(2s_1+2s_2-2)\epsilon_0}{16n} \\ &\qquad + \left(1+\frac{36\gamma n}{\beta^2}\right) \cdot (2s_1)(2s_2) \cdot \left(\frac{\epsilon_0}{16n}\right)^2 \\ &\le \frac{(2s-1)\epsilon_0}{16n} + \frac{36\gamma n}{\beta} \cdot \left(1+\frac{s\epsilon_0}{8n}\right) + \frac{s_1s_2\epsilon_0^2}{32n^2} - \frac{3\epsilon_0}{16n} \qquad (17)\end{aligned}$$

where $s = s_1 + s_2 + 1$ is the number of nodes in g, and where we have used the fact that $36\gamma n \le \beta^2$. We have that

$$\frac{36\gamma n}{\beta} \cdot \left(1 + \frac{s\epsilon_0}{8n}\right) \leq \frac{3 \cdot 36\gamma n}{2\beta} = \frac{3 \cdot 18 \cdot 28 \cdot 5^3 \epsilon_0}{20^5 n} \leq \frac{\epsilon_0}{16n}.$$

Also, since $s_1 + s_2 \leq s \leq 4n$, and since $\epsilon_0 \leq 1/n$,

$$\frac{s_1 s_2 \epsilon_0^2}{32n^2} \leq \frac{4n^2\epsilon_0^2}{32n^2} = \frac{\epsilon_0^2}{8} \leq \frac{\epsilon_0}{8n}.$$

Combined with (17), these facts show that

$$\mathbf{E}\left[|g - \hat{g}|\right] \leq \frac{(2s-1)\epsilon_0}{16n}$$

as desired.

Case 3 *λ is a* LIN *gate.*

In this case, the gate λ is fed by k nodes $\lambda_1, \ldots, \lambda_k$. These nodes subsume subformulas $g_1, \ldots, g_k$, each g_i having canonical form h_i, estimated by $\hat{h}_i$. Let s_i be the number of nodes in g_i. Then by inductive hypothesis,

$$\mathbf{E}\left[|g_i - \hat{g}_i|\right] \leq \frac{(2s_i - 1)\epsilon_0}{16n}$$

where, as usual, $\hat{g}_i = \hat{h}_i / \mathbf{E}\left[\partial f / \partial g_i\right]$.

Let $K = \mathbf{E}\left[\partial f / \partial g\right]$, and suppose that $g = z + \sum w_i g_i$ where z and the w_i's are constants. (If not otherwise specified, it is understood here and below that all summations are for $i = 1, \ldots, k$.) Then

$$\begin{aligned} h_i &= \mathbf{E}\left[\partial f / \partial g_i\right] \cdot g_i \\ &= \mathbf{E}\left[\partial f / \partial g\right] \cdot \mathbf{E}\left[\partial g / \partial g_i\right] \cdot g_i = K w_i g_i. \end{aligned}$$

Thus,

$$\begin{aligned} h &= \mathbf{E}\left[\partial f / \partial g\right] \cdot g = Kg \\ &= Kz + \sum K w_i g_i = Kz + \sum h_i. \end{aligned}$$

Since $\chi(\lambda_i) = \mathbf{E}\left[h_i\right]$ and $\chi(\lambda) = \mathbf{E}\left[h\right]$, this equation implies that

$$Kz = \mathbf{E}\left[h - \sum h_i\right] = \chi(\lambda) - \sum \chi(\lambda_i)$$

and so

$$h = \chi(\lambda) + \sum (h_i - \chi(\lambda_i)).$$

This is the motivation for our algorithm estimating h by

$$\hat{h} = \hat{\chi}(\lambda) + \sum (\hat{h}_i - \hat{\chi}(\lambda_i)).$$

Recall that $\hat{h} = K\hat{g}$ and $\hat{h}_i = Kw_i\hat{g}_i$. Thus,

$$\hat{g} = \frac{\hat{\chi}(\lambda) - \sum \hat{\chi}(\lambda_i)}{K} + \sum w_i \hat{g}_i,$$

and so, letting $s = 1 + \sum s_i$ be the number of nodes in g, we have

$$\begin{aligned}
\mathbf{E}\left[|g - \hat{g}|\right] &\leq \frac{1}{|K|} \cdot \left(\mathbf{E}\left[|\chi(\lambda) - \hat{\chi}(\lambda)|\right] + \sum \mathbf{E}\left[|\chi(\lambda_i) - \hat{\chi}(\lambda_i)|\right]\right) \\
&\quad + \sum |w_i| \cdot \mathbf{E}\left[|g_i - \hat{g}_i|\right] \\
&\leq \frac{18\gamma n(1+k)}{|K|} + \sum \frac{(2s_i - 1)\epsilon_0}{16n} \\
&\leq \frac{(2s-1)\epsilon_0}{16n} + \frac{18\gamma n(1+k)}{\beta} - \frac{(1+k)\epsilon_0}{16n} \\
&\leq \frac{(2s-1)\epsilon_0}{16n}
\end{aligned}$$

as desired.

Combined with the previous cases, this completes the induction, and proves that the final computed hypothesis $\hat{f}$ has error at most $\mathbf{E}\left[\left|f - \hat{f}\right|\right] \leq \epsilon_0/2 \leq \epsilon/2$.

3.5. *Putting it all together*

Thus, we showed in Stage I how to eliminate uncorrelated variables from the target formula f, yielding another formula f_{I} with $\mathbf{E}\left[|f - f_{\mathrm{I}}|\right] \leq \epsilon/2$.

In Stage II, we regarded f_{I} as the target, and showed how to identify gate types, and how to find a skeleton σ' of f_{I} that closely approximates the true skeleton σ.

Finally, in Stage III, we described a procedure for filling in the missing LIN-gate parameters. In particular, we showed that this procedure yields the same result whether applied to the true skeleton σ, or its approximation σ'. Then we showed that when applied to σ, the result $\hat{f}$ is such that $\mathbf{E}\left[\left|f_{\mathrm{I}} - \hat{f}\right|\right] \leq \epsilon/2$. It follows immediately that $\mathbf{E}\left[\left|f - \hat{f}\right|\right] \leq \epsilon$ as desired. In other words, $\hat{f}$ is an ϵ-good model of probability for f with probability at least $1 - \delta$.

All of the operations described are clearly polynomial time, and the sample size is also polynomial. The sample was needed to estimate various expectations. For the accuracy needed, we can use Hoeffding's (1963) Inequality to show that a sample of size $O(\log(n/\delta)/\tau^2)$ suffices.

After the sample is drawn our algorithm records the obtained estimates of these values. All of the remaining operations of the algorithm take negligible time compared to the time needed to compute and record these values from the sample.

Thus we have:

Theorem 1 *There exists an algorithm with the following properties: Given $\epsilon, \delta > 0$ and access to random examples chosen according to a product distribution and classified randomly according to some read-once real formula f, the algorithm outputs an ϵ-good model of probability for f with probability at least $1 - \delta$. The sample size needed by the algorithm is*

$$O\left(\frac{n^{10}}{\epsilon^6} \cdot \left(n + \frac{1}{\epsilon}\right)^2 \cdot \log(n/\delta)\right),$$

and its running time is

$$O\left(\frac{n^{12}}{\epsilon^6} \cdot \left(n + \frac{1}{\epsilon}\right)^2 \cdot \log(n/\delta)\right).$$

From the comments made in Section 2, we have as corollary:

Corollary 1 *There exists a polynomial-time algorithm that PAC-learns the class of read-once Boolean formulas against any product distribution.*

Finally, we remark briefly that the algorithm given in this paper, though analyzed only for Boolean variables, can in fact be modified to handle real-valued variables with range in $[0, 1]$. Note that if x_i is not Boolean, then equation (5) may not hold. However, the given derivation of that equation does show that, in general,

$$C_i = \bar{B}_i \cdot \mathrm{var}(x_i)$$

where $\mathrm{var}(x_i)$ is simply the variance of x_i:

$$\mathrm{var}(x_i) = \mathbf{E}\left[(x_i - \bar{x}_i)^2\right] = \mathbf{E}\left[x_i^2\right] - (\mathbf{E}\left[x_i\right])^2.$$

Similarly, equation (9) becomes

$$T_{ij} = \bar{A}_{ij} \cdot \mathrm{var}(x_i) \cdot \mathrm{var}(x_j).$$

Using these modified versions of equations (5) and (9) we can adjust the rest of the proof appropriately to derive a similar learnability result for non-Boolean variables (with some degradation in the time and sample complexity).

4. Conclusion and open problems

In this paper, we have described a polynomial-time algorithm for learning a class of probabilistic read-once formulas. The algorithm reconstructs an approximation of the formula using empirical estimates of various statistical measures.

The main open question is to determine how far this apparently powerful method can be extended. For example, can such a technique be used to learn read-once formulas over other kinds of gates, such as parity or threshold gates? Can our method be applied to formulas which are not read-once? Can it be extended beyond product distributions? Can it be extended to the so-called two-oracle model? Are there other classes entirely different to which it might be extended, such as decision trees, or finite automata?

Considering the class discussed in this paper, can the algorithm used (or its analysis) be significantly improved? Turning the question around, can we find good, nontrivial lower bounds for such distribution-specific learning problems? It is unclear what such a lower-bound proof would look like, especially since, in the PAC model, much smaller sample sizes are known to suffice in a computationally unbounded setting. (This follows, for instance, from Occam's Razor of Blumer et al. (1987).)

Acknowledgements

Thanks to Ron Rivest for his comments and careful reading of an earlier draft of this material, and to Sally Goldman, Tom Hancock, Lisa Hellerstein and Yishay Mansour for several helpful discussions. I am also grateful to two anonymous referees for their thorough reading and thoughtful suggestions.

This paper was prepared in part while I was at Harvard University with the support of AFOSR Grant 89-0506, and also while I was at the MIT Laboratory for Computer Science with the support of ARO Grant DAAL03-86-K-0171, DARPA Contract N00014-89-J-1988, and a grant from the Siemens Corporation.

References

Angluin, D., Hellerstein, L., and Karpinski, M. (1993). Learning read-once formulas with queries. *Journal of the Association for Computing Machinery*, 40(1):185–210.

Angluin, D. and Laird, P. (1988). Learning from noisy examples. *Machine Learning*, 2(4):343–370.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1987). Occam's razor. *Information Processing Letters*, 24(6):377–380.

Bshouty, N. H., Hancock, T. R., and Hellerstein, L. (1992). Learning arithmetic read-once formulas. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on the Theory of Computing*, pages 370–381.

Furst, M. L., Jackson, J. C., and Smith, S. W. (1991). Improved learning of AC^0 functions. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 317–325.

Goldman, S. A., Kearns, M. J., and Schapire, R. E. (1990). Exact identification of circuits using fixed points of amplification functions. In *31st Annual Symposium on Foundations of Computer Science*, pages 193–202. To appear, *SIAM Journal on Computing*.

Hancock, T. and Hellerstein, L. (1991). Learning read-once formulas over fields and extended bases. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 326–336.

Hancock, T. and Mansour, Y. (1991). Learning monotone $k\mu$ DNF formulas on product distributions. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, pages 179–183.

Hancock, T. R. (1990). Identifying μ-formula decision trees with queries. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 23–37.

Hellerstein, L. and Karpinski, M. (1990). Read-once formulas over different bases. Technical Report 8556-CS, University of Bonn.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30.

Kearns, M., Li, M., Pitt, L., and Valiant, L. (1987). On the learnability of Boolean formulae. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 285–295.

Kearns, M. and Valiant, L. G. (1989). Cryptographic limitations on learning Boolean formulae and finite automata. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 433–444. To appear, *Journal of the Association for Computing Machinery.*

Kearns, M. J. and Schapire, R. E. (1990). Efficient distribution-free learning of probabilistic concepts. In *31st Annual Symposium on Foundations of Computer Science*, pages 382–391. To appear, *Journal of Computer and System Sciences.*

Linial, N., Mansour, Y., and Nisan, N. (1989). Constant depth circuits, Fourier transform, and learnability. In *30th Annual Symposium on Foundations of Computer Science*, pages 574–579.

Pagallo, G. and Haussler, D. (1989). A greedy method for learning μDNF functions under the uniform distribution. Technical Report UCSC-CRL-89-12, University of California Santa Cruz, Computer Research Laboratory.

Sloan, R. H. (1988). Types of noise in data for concept learning. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 91–96.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

Verbeurgt, K. (1990). Learning DNF under the uniform distribution in quasi-polynomial time. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 314–326.

Yamanishi, K. (1992). A learning criterion for stochastic rules. *Machine Learning*, 9(2/3):165–203.